

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2000-59788

(P2000-59788A)

(43)公開日 平成12年2月25日(2000.2.25)

(51)Int.Cl. ⁷	識別記号	F I	テーマコード*(参考)		
H 0 4 N	7/32	H 0 4 N	7/137	Z	5 C 0 5 9
	7/24		7/13	Z	
	7/30		7/133	Z	

審査請求 未請求 請求項の数3 O L (全 62 頁)

(21)出願番号	特願平10-200160	(71)出願人	000002185 ソニー株式会社 東京都品川区北品川6丁目7番35号
(22)出願日	平成10年7月15日(1998.7.15)	(72)発明者	田原 勝己 東京都品川区北品川6丁目7番35号 ソニ ー株式会社内
(31)優先権主張番号	特願平10-58118	(72)発明者	北村 卓也 東京都品川区北品川6丁目7番35号 ソニ ー株式会社内
(32)優先日	平成10年3月10日(1998.3.10)	(74)代理人	100082131 弁理士 稲本 義雄
(33)優先権主張国	日本(J P)		
(31)優先権主張番号	特願平10-157245		
(32)優先日	平成10年6月5日(1998.6.5)		
(33)優先権主張国	日本(J P)		

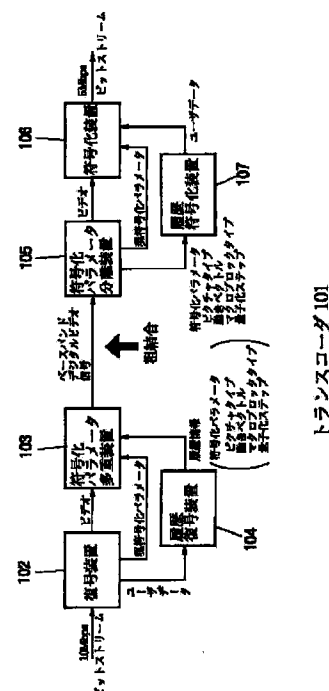
最終頁に続く

(54)【発明の名称】 復号装置および方法、並びに提供媒体

(57)【要約】

【課題】 トランスコーダの規模を小さくするとともに、再符号化に伴う画像の劣化を抑制する。

【解決手段】 符号化パラメータ多重装置103は、現符号化パラメータと履歴復号装置104より供給される履歴情報に含まれる複数世代の符号化パラメータとを、復号装置102より供給されるビデオデータに多重化し、ベースバンドのデジタルビデオ信号として符号化パラメータ分離装置105に出力する。符号化パラメータ分離装置105は、符号化装置106で符号化に使用する符号化パラメータを選択し、現符号化パラメータとして符号化装置106に出力するとともに、残りの複数世代の符号化パラメータを履歴符号化装置107に出力する。符号化装置106は、符号化パラメータ分離装置105より供給されるビデオデータを現符号化パラメータで符号化してビットストリームを生成すると共に、そのビットストリームに履歴符号化装置107より供給される複数世代の符号化パラメータが履歴情報として含まれているユーザデータを多重化し、後段のトランスコーダに出力する。



【特許請求の範囲】

【請求項1】 MPEG規格に基づいて符号化されているビットストリームを復号する復号装置において、前記ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号手段と、前記ビットストリームからビデオデータを復号するビデオデータ復号手段とを備えることを特徴とする復号装置。

【請求項2】 MPEG規格に基づいて符号化されているビットストリームを復号する復号装置の復号方法において、前記ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、前記ビットストリームからビデオデータを復号するビデオデータ復号ステップとを含むことを特徴とする復号方法。

【請求項3】 MPEG規格に基づいて符号化されているビットストリームを復号する復号装置に、前記ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、前記ビットストリームからビデオデータを復号するビデオデータ復号ステップとを含む処理を実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする提供媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、復号装置および方法、並びに提供媒体に関し、特に、動画像信号を、例えば光磁気ディスクや磁気テープなどの記録媒体に記録し、これを再生して、ステレオ視が可能なディスプレイなどに表示したり、テレビ会議システム、テレビ電話システム、放送用機器など、動画像信号を伝送路を介して送信側から受信側に伝送し、受信側において、これを受信して表示する場合などに用いて好適な復号装置および方法、並びに提供媒体に関する。

【0002】

【従来の技術】例えば、テレビ会議システム、テレビ電話システムなどのように、動画像信号を遠隔地に伝送するシステムにおいては、伝送路を効率良く利用するため、映像信号のライン相関やフレーム間相関が利用されて、画像信号が圧縮符号化される。

【0003】画像信号が圧縮符号化される場合、生成されるビットストリームが、所定のビットレートになるように符号化が行われる。しかしながら、実運用上において、伝送路の都合により、ビットストリームのビットレートを変換する必要があることがある。このような場合、図68に示すようなトランスコーダ131により、

符号化されている情報を一旦復号し、ビットレートが所定の値になるように、再び符号化する方法が一般的である。図68の例の場合、10Mbpsで送られてきたビットストリームが、復号装置132により復号され、デジタルビデオ信号として符号化装置133に供給され、符号化装置133により、ビットレートが5Mbpsであるビットストリームに符号化されて出力される。

【0004】

【発明が解決しようとする課題】このように映像信号を再符号化する場合、符号化装置133には、図69に示すように、映像信号のライン相関やフレーム間相関を検出する動き検出部134が必要となり、符号化装置133の規模が大きくなる課題があった。

【0005】また、例えば放送局においては、映像の編集が秒単位で行われるので、フレームの画像情報が他のフレームの画像情報と独立しているほうがよい。そこで、図70に示すように、低いビットレート（3乃至9Mbps）で転送しても画質が劣化しないように、情報が相関関係にあるフレームの集合であるGOP(Group of Picture)を構成するフレーム数が多いLong GOPの符号化装置133-1から出力されたビットストリームは、放送局の符号化装置133-2により、GOPを構成するフレーム数が少ないShort GOPに変換されて高ビットレート（18乃至50Mbps）で伝送され、編集終了後、符号化装置133-3により、再度Long GOPに変換されて出力される。このように、画像情報に符号化、復号が繰り返されると、符号化の度に使用される符号化パラメータが変化するので画像情報が劣化する課題があった。

【0006】本発明はこのような状況に鑑みてなされたものであり、過去に演算した動きベクトルを用いて再符号化を行うことにより、装置の規模を小さくするとともに、再符号化に伴う画像の劣化を抑制することを可能とするものである。

【0007】

【課題を解決するための手段】請求項1に記載の復号装置は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号手段と、ビットストリームからビデオデータを復号するビデオデータ復号手段とを備えることを特徴とする。

【0008】請求項2に記載の復号方法は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、ビットストリームからビデオデータを復号するビデオデータ復号ステップとを含むことを特徴とする。

【0009】請求項3に記載の提供媒体は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号ステップと、ビットストリームからビデ

オデータを復号するビデオデータ復号ステップとを含む処理を実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする。

【0010】請求項1に記載の復号装置、請求項2に記載の復号方法、および請求項3に記載の提供媒体においては、ビットストリームのピクチャ層のユーザデータエリアに挿入された、符号化履歴情報が復号される。

【0011】

【発明の実施の形態】以下に本発明の実施の形態を説明するが、特許請求の範囲に記載の発明の各手段と以下の実施の形態との対応関係を明らかにするために、各手段の後の括弧内に、対応する実施の形態（但し一例）を付加して本発明の特徴を記述すると、次のようになる。

【0012】請求項1に記載の復号装置は、ビットストリームのピクチャ層のユーザデータエリアに挿入された、過去の符号化処理における符号化履歴情報を復号する履歴情報復号手段（例えば、図15の履歴復号装置104）と、ビットストリームからビデオデータを復号するビデオデータ復号手段（例えば、図15の復号装置102）とを備えることを特徴とする。

【0013】但し勿論この記載は、各手段を記載したものに限定することを意味するものではない。

【0014】本発明を適用したトランスコードについて説明する前に、動画像信号の圧縮符号化について説明する。なお、本明細書においてシステムの用語は、複数の装置、手段などにより構成される全体的な装置を意味するものである。

【0015】例えば、テレビ会議システム、テレビ電話システムなどのように、動画像信号を遠隔地に伝送するシステムにおいては、伝送路を効率良く利用するため、映像信号のライン相関やフレーム間相関を利用して、画像信号を圧縮符号化するようになされている。

【0016】ライン相関を利用すると、画像信号を、例えばDCT（離散コサイン変換）処理するなどして圧縮することができる。

【0017】また、フレーム間相関を利用すると、画像信号をさらに圧縮して符号化することが可能となる。例えば図1に示すように、時刻 t_1 乃至 t_3 において、フレーム画像PC1乃至PC3がそれぞれ発生している場合、フレーム画像PC1およびPC2の画像信号の差を演算して、PC12を生成し、また、フレーム画像PC2およびPC3の差を演算して、PC23を生成する。通常、時間的に隣接するフレームの画像は、それ程大きな変化を有していないため、両者の差を演算すると、その差分信号は小さな値のものとなる。そこで、この差分信号を符号化すれば、符号量を圧縮することができる。

【0018】しかしながら、差分信号のみを伝送したのでは、元の画像を復元することができない。そこで、各フレームの画像を、Iピクチャ、PピクチャまたはBピクチャの3種類のピクチャタイプのいずれかとし、画像

信号を圧縮符号化するようにしている。

【0019】すなわち、例えば図2に示すように、フレームF1乃至F17までの17フレームの画像信号をグループオブピクチャ(GOP)とし、処理の1単位とする。そして、その先頭のフレームF1の画像信号はIピクチャとして符号化し、第2番目のフレームF2はBピクチャとして、また第3番目のフレームF3はPピクチャとして、それぞれ処理する。以下、第4番目以降のフレームF4乃至F17は、BピクチャまたはPピクチャとして交互に処理する。

【0020】Iピクチャの画像信号としては、その1フレーム分の画像信号をそのまま伝送する。これに対して、Pピクチャの画像信号としては、基本的には、図2に示すように、それより時間的に先行するIピクチャまたはPピクチャの画像信号からの差分を伝送する。さらにBピクチャの画像信号としては、基本的には、図3に示すように、時間的に先行するフレームまたは後行するフレームの両方の平均値からの差分を求め、その差分を符号化する。

【0021】図4は、このようにして、動画像信号を符号化する方法の原理を示している。同図に示すように、最初のフレームF1は、Iピクチャとして処理されるため、そのまま伝送データF1Xとして伝送路に伝送される（画像内符号化）。これに対して、第2のフレームF2は、Bピクチャとして処理されるため、時間的に先行するフレームF1と、時間的に後行するフレームF3の平均値との差分が演算され、その差分が伝送データF2Xとして伝送される。

【0022】ただし、このBピクチャとしての処理は、さらに細かく説明すると、4種類存在する。その第1の処理は、元のフレームF2のデータをそのまま伝送データF2Xとして伝送するものであり（SP1）（イントラ符号化）、Iピクチャにおける場合と同様の処理となる。第2の処理は、時間的に後のフレームF3からの差分を演算し、その差分（SP2）を伝送するものである（後方予測符号化）。第3の処理は、時間的に先行するフレームF1との差分（SP3）を伝送するものである（前方予測符号化）。さらに第4の処理は、時間的に先行するフレームF1と後行するフレームF3の平均値との差分（SP4）を生成し、これを伝送データF2Xとして伝送するものである（両方向予測符号化）。

【0023】実際には、上述した4つの方法のうちの伝送データが最も少なくなる方法が採用される。

【0024】なお、差分データを伝送するとき、差分を演算する対象となるフレームの画像（予測画像）との間の動きベクトル x_1 （フレームF1とF2の間の動きベクトル）（前方予測の場合）、もしくは x_2 （フレームF3とF2の間の動きベクトル）（後方予測の場合）、または x_1 と x_2 の両方（両方向予測の場合）が、差分データとともに伝送される。

【0025】また、PピクチャのフレームF3は、時間的に先行するフレームF1を予測画像として、このフレームとの差分信号（SP3）と、動きベクトルx3が演算され、これが伝送データF3Xとして伝送される（前方予測符号化）。あるいはまた、元のフレームF3のデータが、そのままデータF3Xとして伝送される（SP1）（イントラ符号化）。いずれの方法により伝送されるかは、Bピクチャにおける場合と同様に、伝送データがより少なくなる方法が選択される。

【0026】図5は、上述した原理に基づいて、動画画像信号を符号化して伝送し、これを復号化する装置の構成例を示している。符号化装置1は、入力された映像信号を符号化し、伝送路としての記録媒体3に伝送するようになされている。そして、復号装置2は、記録媒体3に記録された信号を再生し、これを復号して出力するようになされている。

【0027】符号化装置1においては、入力された映像信号が前処理回路11に輸入され、そこで輝度信号と色信号（本実施の形態の場合、色差信号）が分離され、それぞれA/D変換器12、13でアナログ信号がデジタル信号に変換される。A/D変換器12、13によりデジタル信号に変換された映像信号は、フレームメモリ14に供給され、記憶される。フレームメモリ14は、輝度信号を輝度信号フレームメモリ15に、また、色差信号を色差信号フレームメモリ16に、それぞれ記憶させる。

【0028】フォーマット変換回路17は、フレームメモリ14に記憶されたフレームフォーマットの信号を、ブロックフォーマットの信号に変換する。すなわち、図6に示すように、フレームメモリ14に記憶された映像信号は、1ライン当りHドットのラインがVライン集められた、図6(A)に示すようなフレームフォーマットのデータとされている。フォーマット変換回路17は、この1フレームの信号を、図6(B)に示すように、16ラインを単位としてM個のスライスに区分する。そして、各スライスは、M個のマクロブロックに分割される。マクロブロックは、図6(C)に示すように、16×16個の画素（ドット）に対応する輝度信号により構成され、この輝度信号は、さらに8×8ドットを単位とするブロックY[1]乃至Y[4]に区分される。そして、この16×16ドットの輝度信号には、8×8ドットのCb信号と、8×8ドットのCr信号が対応される。

【0029】このように、ブロックフォーマットに変換されたデータは、フォーマット変換回路17からエンコーダ18に供給され、ここでエンコード（符号化）が行われる。その詳細については、図7を参照して後述する。

【0030】エンコーダ18によりエンコードされた信号は、ビットストリームとして伝送路に出力される。例えば記録回路19に供給され、デジタル信号として記録媒体3に記録される。

【0031】再生回路30により記録媒体3より再生されたデータは、復号装置2のデコーダ31に供給され、デコードされる。デコーダ31の詳細については、図12を参照して後述する。

【0032】デコーダ31によりデコードされたデータは、フォーマット変換回路32に輸入され、ブロックフォーマットからフレームフォーマットに変換される。そして、フレームフォーマットの輝度信号は、フレームメモリ33の輝度信号フレームメモリ34に供給されて記憶され、色差信号は色差信号フレームメモリ35に供給されて記憶される。輝度信号フレームメモリ34と色差信号フレームメモリ35から読み出された輝度信号と色差信号は、それぞれD/A変換器36、37によりアナログ信号に変換され、後処理回路38に供給される。後処理回路38は、輝度信号と色差信号を合成して出力する。

【0033】次に図7を参照して、エンコーダ18の構成について説明する。符号化される画像データは、マクロブロック単位で動きベクトル検出回路50に輸入される。動きベクトル検出回路50は、予め設定されている所定のシーケンスに従って、各フレームの画像データを、Iピクチャ、Pピクチャ、またはBピクチャとして処理する。シーケンスに入力される各フレームの画像を、I、P、またはBのいずれのピクチャとして処理するかは、予め定められている（例えば、図2と図3に示したように、フレームF1乃至F17により構成されるグループオブピクチャが、I、B、P、B、P、・・・B、Pとして処理される）。

【0034】Iピクチャとして処理されるフレーム（例えば、フレームF1）の画像データは、動きベクトル検出回路50からフレームメモリ51の前方原画像部51aに転送、記憶され、Bピクチャとして処理されるフレーム（例えば、フレームF2）の画像データは、原画像部51bに転送、記憶され、Pピクチャとして処理されるフレーム（例えば、フレームF3）の画像データは、後方原画像部51cに転送、記憶される。

【0035】また、次のタイミングにおいて、さらにBピクチャ（フレームF4）またはPピクチャ（フレームF5）として処理すべきフレームの画像が輸入されたとき、それまで後方原画像部51cに記憶されていた最初のPピクチャ（フレームF3）の画像データが、前方原画像部51aに転送され、次のBピクチャ（フレームF4）の画像データが、原画像部51bに記憶（上書き）され、次のPピクチャ（フレームF5）の画像データが、後方原画像部51cに記憶（上書き）される。このような動作が順次繰り返される。

【0036】フレームメモリ51に記憶された各ピクチャの信号は、そこから読み出され、予測モード切り替え回路52において、フレーム予測モード処理、またはフィールド予測モード処理が行われる。

【0037】さらにまた、予測判定回路54の制御の下に、演算部53において、画像内予測、前方予測、後方予測、または両方向予測の演算が行なわれる。これらの処理のうち、いずれの処理を行なうかは、予測誤差信号（処理の対象とされている参照画像と、これに対する予測画像との差分）に対応して決定される。このため、動きベクトル検出回路50は、この判定に用いられる予測誤差信号の絶対値和（自乗和でもよい）を生成する。

【0038】ここで、予測モード切り替え回路52におけるフレーム予測モードとフィールド予測モードについて説明する。

【0039】フレーム予測モードが設定された場合においては、予測モード切り替え回路52は、動きベクトル検出回路50より供給される4個の輝度ブロックY

[1]乃至Y[4]を、そのまま後段の演算部53に出力する。すなわち、この場合においては、図8に示すように、各輝度ブロックに奇数フィールドのラインのデータと、偶数フィールドのラインのデータとが混在した状態となっている。このフレーム予測モードにおいては、4個の輝度ブロック（マクロブロック）を単位として予測が行われ、4個の輝度ブロックに対して1個の動きベクトルが対応される。

【0040】これに対して、予測モード切り替え回路52は、フィールド予測モードにおいては、図8に示す構成で動きベクトル検出回路50より入力される信号を、図9に示すように、4個の輝度ブロックのうち、輝度ブロックY[1]とY[2]を、例えば奇数フィールドのラインのドットだけで構成させ、他の2個の輝度ブロックY[3]とY[4]を、偶数フィールドのラインのドットだけで構成させて、演算部53に出力する。この場合においては、2個の輝度ブロックY[1]とY[2]に対して、1個の動きベクトルが対応され、他の2個の輝度ブロックY[3]とY[4]に対して、他の1個の動きベクトルが対応される。

【0041】動きベクトル検出回路50は、フレーム予測モードにおける予測誤差の絶対値和、およびフィールド予測モードにおける予測誤差の絶対値和を予測モード切り替え回路52に出力する。予測モード切り替え回路52は、フレーム予測モードとフィールド予測モードにおける予測誤差の絶対値和を比較し、その値が小さい予測モードに対応する処理を施して、データを演算部53に出力する。

【0042】ただし、このような処理は、実際には動きベクトル検出回路50で行われる。すなわち、動きベクトル検出回路50は、決定されたモードに対応する構成の信号を予測モード切り替え回路52に出力し、予測モード切り替え回路52は、その信号を、そのまま後段の演算部53に出力する。

【0043】なお、色差信号は、フレーム予測モードの場合、図8に示すように、奇数フィールドのラインのデ

ータと偶数フィールドのラインのデータとが混在する状態で、演算部53に供給される。また、フィールド予測モードの場合、図9に示すように、各色差ブロックC

b, Crの上半分（4ライン）が、輝度ブロックY

[1], Y[2]に対応する奇数フィールドの色差信号とされ、下半分（4ライン）が、輝度ブロックY

[3], Y[4]に対応する偶数フィールドの色差信号とされる。

【0044】また、動きベクトル検出回路50は、以下に示すようにして、予測判定回路54において、画像内予測、前方予測、後方予測、または両方向予測のいずれの予測を行なうかを決定するための予測誤差の絶対値和を生成する。

【0045】すなわち、画像内予測の予測誤差の絶対値和として、参照画像のマクロブロックの信号Aijの総和 $\sum Aij$ の絶対値 $|\sum Aij|$ と、マクロブロックの信号Aijの絶対値 $|Aij|$ の総和 $\sum |Aij|$ の差を求める。また、前方予測の予測誤差の絶対値和として、参照画像のマクロブロックの信号Aijと、予測画像のマクロブロックの信号Bijの差 $Aij - Bij$ の絶対値 $|Aij - Bij|$ の総和 $\sum |Aij - Bij|$ を求める。また、後方予測と両方向予測の予測誤差の絶対値和も、前方予測における場合と同様に（その予測画像を前方予測における場合と異なる予測画像に変更して）求める。

【0046】これらの絶対値和は、予測判定回路54に供給される。予測判定回路54は、前方予測、後方予測および両方向予測の予測誤差の絶対値和のうちの最も小さいものを、インタ予測の予測誤差の絶対値和として選択する。さらに、このインタ予測の予測誤差の絶対値和と、画像内予測の予測誤差の絶対値和とを比較し、その小さい方を選択し、この選択した絶対値和に対応するモードを予測モードとして選択する。すなわち、画像内予測の予測誤差の絶対値和の方が小さければ、画像内予測モードが設定される。インタ予測の予測誤差の絶対値和の方が小さければ、前方予測、後方予測または両方向予測モードのうちの対応する絶対値和が最も小さかったモードが設定される。

【0047】このように、動きベクトル検出回路50は、参照画像のマクロブロックの信号を、フレームまたはフィールド予測モードのうち、予測モード切り替え回路52により選択されたモードに対応する構成で、予測モード切り替え回路52を介して演算部53に供給するとともに、4つの予測モードのうちの予測判定回路54により選択された予測モードに対応する予測画像と参照画像の間の動きベクトルを検出し、可変長符号化回路58と動き補償回路64に出力する。上述したように、この動きベクトルとしては、対応する予測誤差の絶対値和が最小となるものが選択される。

【0048】予測判定回路54は、動きベクトル検出回路50が前方原画像部51aより1ピクチャの画像デー

タを読み出しているとき、予測モードとして、フレームまたはフィールド（画像）内予測モード（動き補償を行わないモード）を設定し、演算部53のスイッチ53dを接点a側に切り替える。これにより、1ピクチャの画像データがDCTモード切り替え回路55に入力される。

【0049】DCTモード切り替え回路55は、図10または図11に示すように、4個の輝度ブロックのデータを、奇数フィールドのラインと偶数フィールドのラインが混在する状態（フレームDCTモード）、または、分離された状態（フィールドDCTモード）、のいずれかの状態にして、DCT回路56に出力する。

【0050】すなわち、DCTモード切り替え回路55は、奇数フィールドと偶数フィールドのデータを混在してDCT処理した場合における符号化効率と、分離した状態においてDCT処理した場合の符号化効率とを比較し、符号化効率の良好なモードを選択する。

【0051】例えば、入力された信号を、図10に示すように、奇数フィールドと偶数フィールドのラインが混在する構成とし、上下に隣接する奇数フィールドのラインの信号と偶数フィールドのラインの信号の差を演算し、さらにその絶対値の和（または自乗和）を求める。

【0052】また、入力された信号を、図11に示すように、奇数フィールドと偶数フィールドのラインが分離した構成とし、上下に隣接する奇数フィールドのライン同士の信号の差と、偶数フィールドのライン同士の信号の差を演算し、それぞれの絶対値の和（または自乗和）を求める。

【0053】さらに、両者（絶対値和）を比較し、小さい値に対応するDCTモードを設定する。すなわち、前者の方が小さければ、フレームDCTモードを設定し、後者の方が小さければ、フィールドDCTモードを設定する。

【0054】そして、選択したDCTモードに対応する構成のデータをDCT回路56に出力するとともに、選択したDCTモードを示すDCTフラグを、可変長符号化回路58、および動き補償回路64に出力する。

【0055】予測モード切り替え回路52における予測モード（図8と図9）と、このDCTモード切り替え回路55におけるDCTモード（図10と図11）を比較して明らかなように、輝度ブロックに関しては、両者の各モードにおけるデータ構造は実質的に同一である。

【0056】予測モード切り替え回路52において、フレーム予測モード（奇数ラインと偶数ラインが混在するモード）が選択された場合、DCTモード切り替え回路55においても、フレームDCTモード（奇数ラインと偶数ラインが混在するモード）が選択される可能性が高く、また予測モード切り替え回路52において、フィールド予測モード（奇数フィールドと偶数フィールドのデータが分離されたモード）が選択された場合、DCTモード切り替え回路55において、フィールドDCTモード（奇数フィールドと偶数フィールドのデータが分離されたモー

ド）が選択される可能性が高い。

【0057】しかしながら、必ずしも常にこのようにモードが選択されるわけではなく、予測モード切り替え回路52においては、予測誤差の絶対値和が小さくなるようにモードが決定され、DCTモード切り替え回路55においては、符号化効率が良好となるようにモードが決定される。

【0058】DCTモード切り替え回路55より出力された1ピクチャの画像データは、DCT回路56に入力されてDCT処理され、DCT係数に変換される。このDCT係数は、量子化回路57に入力され、送信バッファ59のデータ蓄積量（バッファ蓄積量）に対応した量子化スケールで量子化された後、可変長符号化回路58に入力される。

【0059】可変長符号化回路58は、量子化回路57より供給される量子化スケール（スケール）に対応して、量子化回路57より供給される画像データ（いまの場合、1ピクチャのデータ）を、例えばハフマン符号などの可変長符号に変換し、送信バッファ59に出力する。

【0060】可変長符号化回路58にはまた、量子化回路57より量子化スケール（スケール）、予測判定回路54より予測モード（画像内予測、前方予測、後方予測、または両方向予測のいずれが設定されたかを示すモード）、動きベクトル検出回路50より動きベクトル、予測モード切り替え回路52より予測フラグ（フレーム予測モードまたはフィールド予測モードのいずれが設定されたかを示すフラグ）、およびDCTモード切り替え回路55が出力するDCTフラグ（フレームDCTモードまたはフィールドDCTモードのいずれが設定されたかを示すフラグ）が入力されており、これらも可変長符号化される。

【0061】送信バッファ59は、入力されたデータを一時蓄積し、蓄積量に対応するデータを量子化回路57に出力する。送信バッファ59は、そのデータ残量が許容上限値まで増量すると、量子化制御信号によって量子化回路57の量子化スケールを大きくすることにより、量子化データのデータ量を低下させる。また、これとは逆に、データ残量が許容下限値まで減少すると、送信バッファ59は、量子化制御信号によって量子化回路57の量子化スケールを小さくすることにより、量子化データのデータ量を増大させる。このようにして、送信バッファ59のオーバフローまたはアンダフローが防止される。

【0062】そして、送信バッファ59に蓄積されたデータは、所定のタイミングで読み出され、伝送路に出力され、例えば記録回路19を介して記録媒体3に記録される。

【0063】一方、量子化回路57より出力された1ピクチャのデータは、逆量子化回路60に入力され、量子

化回路57より供給される量子化スケールに対応して逆量子化される。逆量子化回路60の出力は、IDCT（逆離散コサイン変換）回路61に入力され、逆離散コサイン変換処理された後、演算器62を介してフレームメモリ63の前方予測画像部63a供給されて記憶される。

【0064】動きベクトル検出回路50は、シーケンシャルに入力される各フレームの画像データを、たとえば、I, B, P, B, P, B...のピクチャとしてそれぞれ処理する場合、最初に入力されたフレームの画像データをIピクチャとして処理した後、次に入力されたフレームの画像をBピクチャとして処理する前に、さらにその次に入力されたフレームの画像データをPピクチャとして処理する。Bピクチャは、後方予測を伴うため、後方予測画像としてのPピクチャが先に用意されていないと、復号することができないからである。

【0065】そこで動きベクトル検出回路50は、Iピクチャの処理の次に、後方原画像部51cに記憶されているPピクチャの画像データの処理を開始する。そして、上述した場合と同様に、マクロブロック単位でのフレーム間差分（予測誤差）の絶対値和が、動きベクトル検出回路50から予測モード切り替え回路52と予測判定回路54に供給される。予測モード切り替え回路52と予測判定回路54は、このPピクチャのマクロブロックの予測誤差の絶対値和に対応して、フレーム／フィールド予測モード、または画像内予測、前方予測、後方予測、もしくは両方向予測の予測モードを設定する。

【0066】演算部53は、画像内予測モードが設定されたとき、スイッチ53dを上述したように接点a側に切り替える。したがって、このデータは、Iピクチャのデータと同様に、DCTモード切り替え回路55、DCT回路56、量子化回路57、可変長符号化回路58、および送信バッファ59を介して伝送路に伝送される。また、このデータは、逆量子化回路60、IDCT回路61、および演算器62を介してフレームメモリ63の後方予測画像部63bに供給されて記憶される。

【0067】また、前方予測モードが設定された場合、スイッチ53dが接点bに切り替えられるとともに、フレームメモリ63の前方予測画像部63aに記憶されている画像（いまの場合、Iピクチャの画像）データが読み出され、動き補償回路64により、動きベクトル検出回路50が出力する動きベクトルに対応して動き補償される。すなわち、動き補償回路64は、予測判定回路54より前方予測モードの設定が指令されたとき、前方予測画像部63aの読み出しアドレスを、動きベクトル検出回路50が、現在、出力しているマクロブロックの位置に対応する位置から動きベクトルに対応する分だけずらしてデータを読み出し、予測画像データを生成する。

【0068】動き補償回路64より出力された予測画像データは、演算器53aに供給される。演算器53aは、予測モード切り替え回路52より供給された参照画

像のマクロブロックのデータから、動き補償回路65より供給された、このマクロブロックに対応する予測画像データを減算し、その差分（予測誤差）を出力する。この差分データは、DCTモード切り替え回路55、DCT回路56、量子化回路57、可変長符号化回路58、および送信バッファ59を介して伝送路に伝送される。また、この差分データは、逆量子化回路60、およびIDCT回路61により局所的に復号され、演算器62に入力される。

【0069】この演算器62にはまた、演算器53aに供給されている予測画像データと同一のデータが供給されている。演算器62は、IDCT回路61が出力する差分データに、動き補償回路64が出力する予測画像データを加算する。これにより、元の（復号した）Pピクチャの画像データが得られる。このPピクチャの画像データは、フレームメモリ63の後方予測画像部63bに供給されて記憶される。

【0070】動きベクトル検出回路50は、このように、IピクチャとPピクチャのデータが前方予測画像部63aと後方予測画像部63bにそれぞれ記憶された後、次にBピクチャの処理を実行する。予測モード切り替え回路52と予測判定回路54は、マクロブロック単位でのフレーム間差分の絶対値和の大きさに対応して、フレーム／フィールドモードを設定し、また、予測モードを画像内予測モード、前方予測モード、後方予測モード、または両方向予測モードのいずれかに設定する。

【0071】上述したように、画像内予測モードまたは前方予測モードの時、スイッチ53dは接点aまたはbに切り替えられる。このとき、Pピクチャにおける場合と同様の処理が行われ、データが伝送される。

【0072】これに対して、後方予測モードまたは両方向予測モードが設定された時、スイッチ53dは、接点cまたはdにそれぞれ切り替えられる。

【0073】スイッチ53dが接点cに切り替えられている後方予測モードの時、後方予測画像部63bに記憶されている画像（いまの場合、Pピクチャの画像）データが読み出され、動き補償回路64により、動きベクトル検出回路50が出力する動きベクトルに対応して動き補償される。すなわち、動き補償回路64は、予測判定回路54より後方予測モードの設定が指令されたとき、後方予測画像部63bの読み出しアドレスを、動きベクトル検出回路50が、現在、出力しているマクロブロックの位置に対応する位置から動きベクトルに対応する分だけずらしてデータを読み出し、予測画像データを生成する。

【0074】動き補償回路64より出力された予測画像データは、演算器53bに供給される。演算器53bは、予測モード切り替え回路52より供給された参照画像のマクロブロックのデータから、動き補償回路64より供給された予測画像データを減算し、その差分を出力

する。この差分データは、DCTモード切り替え回路55、DCT回路56、量子化回路57、可変長符号化回路58、および送信バッファ59を介して伝送路に伝送される。

【0075】スイッチ53dが接点dに切り替えられている両方向予測モードの時、前方予測画像部63aに記憶されている画像（いまの場合、Iピクチャの画像）データと、後方予測画像部63bに記憶されている画像（いまの場合、Pピクチャの画像）データが読み出され、動き補償回路64により、動きベクトル検出回路50が出力する動きベクトルに対応して動き補償される。

【0076】すなわち、動き補償回路64は、予測判定回路54より両方向予測モードの設定が指令されたとき、前方予測画像部63aと後方予測画像部63bの読み出しアドレスを、動きベクトル検出回路50がいま出力しているマクロブロックの位置に対応する位置から動きベクトル（この場合の動きベクトルは、前方予測画像用と後方予測画像用の2つとなる）に対応する分だけずらしてデータを読み出し、予測画像データを生成する。

【0077】動き補償回路64より出力された予測画像データは、演算器53cに供給される。演算器53cは、動きベクトル検出回路50より供給された参照画像のマクロブロックのデータから、動き補償回路64より供給された予測画像データの平均値を減算し、その差分を出力する。この差分データは、DCTモード切り替え回路55、DCT回路56、量子化回路57、可変長符号化回路58、および送信バッファ59を介して伝送路に伝送される。

【0078】Bピクチャの画像は、他の画像の予測画像とされることがないため、フレームメモリ63には記憶されない。

【0079】なお、フレームメモリ63において、前方予測画像部63aと後方予測画像部63bは、必要に応じてバンク切り替えが行われ、所定の参照画像に対して、一方または他方に記憶されているものを、前方予測画像あるいは後方予測画像として切り替えて出力することができる。

【0080】上述した説明においては、輝度ブロックを中心として説明をしたが、色差ブロックについても同様に、図8乃至図11に示すマクロブロックを単位として処理されて伝送される。なお、色差ブロックを処理する場合の動きベクトルは、対応する輝度ブロックの動きベクトルを垂直方向と水平方向に、それぞれ1/2にしたものが用いられる。

【0081】図12は、図5のデコーダ31の構成を示すブロック図である。伝送路（記録媒体3）を介して伝送された符号化された画像データは、図示せぬ受信回路で受信されたり、再生装置で再生され、受信バッファ81に一時記憶された後、復号回路90の可変長復号化回路82に供給される。可変長復号化回路82は、受信バ

ッファ81より供給されたデータを可変長復号化し、動きベクトル、予測モード、予測フラグ、およびDCTフラグを動き補償回路87に出力し、量子化スケールを逆量子化回路83に出力するとともに、復号された画像データを逆量子化回路83に出力する。

【0082】逆量子化回路83は、可変長復号化回路82より供給された画像データを、同じく可変長復号化回路82より供給された量子化スケールに従って逆量子化し、IDCT回路84に出力する。逆量子化回路83より出力されたデータ（DCT係数）は、IDCT回路84により、逆離散コサイン変換処理が施され、演算器85に供給される。

【0083】IDCT回路84より演算器85に供給された画像データが、Iピクチャのデータである場合、そのデータは演算器85より出力され、演算器85に後に入力される画像データ（PまたはBピクチャのデータ）の予測画像データ生成のために、フレームメモリ86の前方予測画像部86aに供給されて記憶される。また、このデータは、フォーマット変換回路32（図5）に出力される。

【0084】IDCT回路84より供給された画像データが、その1フレーム前の画像データを予測画像データとするPピクチャのデータであり、前方予測モードのデータである場合、フレームメモリ86の前方予測画像部86aに記憶されている、1フレーム前の画像データ（Iピクチャのデータ）が読み出され、動き補償回路87で可変長復号化回路82より出力された動きベクトルに対応する動き補償が施される。そして、演算器85において、IDCT回路84より供給された画像データ（差分のデータ）と加算され、出力される。この加算されたデータ、すなわち、復号されたPピクチャのデータは、演算器85に後に入力される画像データ（BピクチャまたはPピクチャのデータ）の予測画像データ生成のために、フレームメモリ86の後方予測画像部86bに供給されて記憶される。

【0085】Pピクチャのデータであっても、画像内予測モードのデータは、Iピクチャのデータと同様に、演算器85において処理は行われず、そのまま後方予測画像部86bに記憶される。

【0086】このPピクチャは、次のBピクチャの次に表示されるべき画像であるため、この時点では、まだフォーマット変換回路32へ出力されない（上述したように、Bピクチャの後に入力されたPピクチャが、Bピクチャより先に処理され、伝送されている）。

【0087】IDCT回路84より供給された画像データが、Bピクチャのデータである場合、可変長復号化回路82より供給された予測モードに対応して、フレームメモリ86の前方予測画像部86aに記憶されているIピクチャの画像データ（前方予測モードの場合）、後方予測画像部86bに記憶されているPピクチャの画像デー

タ（後方予測モードの場合）、または、その両方の画像データ（両方向予測モードの場合）が読み出され、動き補償回路87において、可変長復号化回路82より出力された動きベクトルに対応する動き補償が施されて、予測画像が生成される。但し、動き補償を必要としない場合（画像内予測モードの場合）、予測画像は生成されない。

【0088】このようにして、動き補償回路87で動き補償が施されたデータは、演算器85において、IDCT回路84の出力と加算される。この加算出力は、フォーマット変換回路32に出力される。

【0089】ただし、この加算出力はBピクチャのデータであり、他の画像の予測画像生成のために利用されることがないため、フレームメモリ86には記憶されない。

【0090】Bピクチャの画像が出力された後、後方予測画像部86bに記憶されているPピクチャの画像データが読み出され、動き補償回路87を介して演算器85に供給される。但し、このとき、動き補償は行われない。

【0091】なお、このデコーダ31には、図5のエンコーダ18における予測モード切り替え回路52とDCTモード切り替え回路55に対応する回路が図示されていないが、これらの回路に対応する処理、すなわち、奇数フィールドと偶数フィールドのラインの信号が分離された構成を元の構成に必要なに応じて戻す処理は、動き補償回路87により実行される。

【0092】また、上述した説明においては、輝度信号の処理について説明したが、色差信号の処理も同様に行われる。ただし、この場合の動きベクトルは、輝度信号用の動きベクトルを、垂直方向および水平方向に1/2にしたものが用いられる。

【0093】図13は、符号化された画像の品質を示している。画像の品質(SNR:Signal to Noise Ratio)は、ピクチャタイプに対応して制御され、Iピクチャ、およびPピクチャは高品質とされ、Bピクチャは、I、Pピクチャに比べて劣る品質とされて伝送される。これは、人間の視覚特性を利用した手法であり、全ての画像品質を平均化するよりも、品質を振動させたほうが視覚上の画質が良くなるためである。このピクチャタイプに対応した画質の制御は、図7の量子化回路57により実行される。

【0094】図14は、本発明を適用したトランスコーダ101の構成を示しており、図15は、そのさらに詳細な構成を示している。復号装置102は、所定のビットレート（この例の場合、10Mbps）のビットストリームに含まれる（多重化されている）符号化された画像信号を、ビットストリームに含まれる（多重化されている）そのビットストリームの現符号化パラメータ（フレーム／フィールドDCTフラグ、フレーム／フィールド予測フ

ラグ、予測モード、ピクチャタイプ、動きベクトル、マクロブロック情報、および量子化スケール）を用いて復号し、符号化パラメータ多重装置103に出力するとともに、現符号化パラメータも符号化パラメータ多重装置103に出力するようになされている。

【0095】復号装置102はまた、ビットストリームに含まれるユーザデータを復号、分離し、履歴復号装置104に出力する。その詳細は後述するが、このユーザデータには、直近の3世代分の符号化パラメータで構成される世代履歴情報が含まれている。これに対して、現符号化パラメータは、例えばgroup_of_pictures_header(1), extension_and_user_data(1), picture_header(), picture_coding_extension(), extensions_data(2), picture_data(), または、sequence_extension()に含まれている（後述する図38）。履歴復号装置104は、入力されたユーザデータを復号し、3世代分の符号化パラメータを含む世代履歴情報を符号化パラメータ多重装置103に出力する。

【0096】なお、復号装置102は、図5の復号装置2のデコーダ31（図12）を図16に示すデコーダ111に変更したものである。デコーダ111の可変長復号化回路112は、現符号化パラメータをビットストリームから抽出し、所定の回路に供給するとともに、世代履歴情報を含むユーザデータを抽出し、履歴復号装置104に出力するようになされている。デコーダ111のその他の構成は、デコーダ31と同様であるので、その説明は省略する。

【0097】符号化パラメータ多重装置103は、復号された画像データの空き領域（その詳細は、図18を参照して説明する）に4世代分の符号化パラメータを書き込み（多重化し）、ベースバンドのデジタルビデオ信号として、粗結合された（符号化パラメータ伝送用の専用バス等が設けられていない）符号化パラメータ分離装置105に出力する。符号化パラメータ分離装置105は、ベースバンドのデジタルビデオ信号から、画像データと、符号化装置106で符号化に用いる符号化パラメータを分離して符号化装置106に供給するようになされている。

【0098】符号化パラメータ分離装置105はまた、入力されたベースバンドのデジタルビデオ信号から、符号化装置106で用いる符号化パラメータを除く3世代分の符号化パラメータを抽出し、履歴符号化装置107に出力する。履歴符号化装置107は、入力された3世代分の符号化パラメータをユーザデータに書き込み、そのユーザデータを符号化装置106に出力する。

【0099】符号パラメータが書き込まれる画像データのフォーマットについて、図17と図18を参照して説明する。1個のマクロブロックは、図17に示すように、16×16画素で構成される。この16×16画素のデータは、8×8画素の輝度信号Y[0][x]乃至Y[4][x]

と、 8×8 画素の色差信号 $Cr[0][x]$ 、 $Cr[1][x]$ および $Cb[0][x]$ 、 $Cb[1][x]$ ($x=2$ 乃至 9) から構成されている。例えば、輝度信号 $Y[0][9]$ は、 8×8 画素の1行目の画素(8画素)の輝度信号を示している。1画素当たりの輝度信号の情報量は8ビットなので、輝度信号 $Y[0][9]$ の情報量は、 8 (画素) $\times 8$ (ビット) $=64$ ビットとなる。色差信号についても同様である。

【0100】これに対して、画像データのフォーマットは、図18に示すように、10行分の領域(D0乃至D9)が設けられているので、2行分の領域(D0, D1)が不要となる。この空き領域には、 64 ビット $\times 16=1024$ ビットの情報が記録できるので、この2行分の領域に本来の画像データ以外の符号化パラメータを書き込む。なお、1個のマクロブロックに対応する符号化パラメータは、 256 ビットの情報量があるので、この領域には、過去4回の符号化に使用された符号化パラメータを記録することができる。

【0101】符号化パラメータ多重装置103から符号化パラメータ分離装置105に伝送される画像データ(デジタルビデオ信号)には、輝度信号 Y 、色差信号 Cr 、 Cb を記載する領域として、10行分(D0乃至D9)の領域が設けられている。しかしながら実際に輝度信号 Y 等が書き込まれる領域は、D2乃至D9の8行分の領域であり、D0, D1の領域は利用されない。そこで、この2ビットの領域を符号化パラメータの書き込み用領域として利用する。これにより、図17の 16×16 画素の所定の位置の画素の下位2ビットに、符号化パラメータが書き込まれることとなる。

【0102】符号化装置106は、これから行う符号化のための符号化パラメータとして供給された現符号化パラメータを利用して画像データを符号化するとともに、履歴符号化装置107から供給されるユーザデータをビットストリームに多重化して、所定のビットレート(この例の場合、5Mbps)でSDTI(Serial Data Transfer Interface)108- i ($i=1, 2, \dots, N$) (後述する図30)に出力するようになされている。

【0103】なお、符号化装置106は、図5の符号化装置1のエンコーダ18(図7)を図19に示すエンコーダ121に変更したものである。エンコーダ121は、エンコーダ18から符号化パラメータを生成する動きベクトル検出回路50、フレームメモリ51、予測モード切り替え回路52、予測判定回路54、およびDCTモード切り替え回路55を削除し、履歴符号化装置107の出力するユーザデータを可変長符号化回路58で可変長符号化するようにしたものである。エンコーダ121のその他の構成は、エンコーダ18と同様であるので、その説明は省略する。

【0104】次に、図15における履歴復号装置104と履歴符号化装置107についてさらに説明する。同図に示すように、履歴復号装置104は、復号装置102

より供給されるユーザデータをデコードするユーザデータデコーダ201、ユーザデータデコーダ201の出力を変換するコンバータ202、およびコンバータ202の出力から履歴情報を再生するヒストリデコーダ203により構成されている。

【0105】また、履歴符号化装置107は、符号化パラメータ分離装置105より供給される3世代分の符号化パラメータをフォーマット化するヒストリフォーマッタ211、ヒストリフォーマッタ211の出力を変換するコンバータ212、コンバータ212の出力をユーザデータのフォーマットにフォーマットするユーザデータフォーマッタ213により構成されている。

【0106】ユーザデータデコーダ201は、復号装置102より供給されるユーザデータをデコードして、コンバータ202に出力する。詳細は後述するが、ユーザデータ(`user_data()`)は、`user_data_start_code`と`user_data`からなり、MPEG規格においては`user_data`の中に、連続する23ビットの"0"を発生させることを禁止している。これは、`start_code`を誤検出されないようにするためである。履歴情報内には、このような連続する23ビット以上の"0"が存在することがあり得るので、これを処理して、`converted_history_stream()`(後述する図38)に変換する必要がある。この変換を行うのは、履歴符号化装置107のコンバータ212である。履歴復号装置104のコンバータ202は、このコンバータ212と逆の変換処理を行うものである。

【0107】ヒストリデコーダ203は、コンバータ202の出力から履歴情報を生成し、符号化パラメータ多重装置103に出力する。

【0108】一方、履歴符号化装置107においては、ヒストリフォーマッタ211が符号化パラメータ分離装置105より供給される3世代分の符号化パラメータを履歴情報のフォーマットに変換する。このフォーマットには、固定長のもの(後述する図40乃至図46)と、可変長のもの(後述する図47)とがある。これらの詳細については後述する。

【0109】ヒストリフォーマッタ211により、フォーマット化された履歴情報は、コンバータ212において、`converted_history_stream()`に変換される。これは、上述したように、`user_data()`の`start_code`が誤検出されないようにするためのものである。すなわち、履歴情報内には連続する23ビット以上の"0"が存在するが、`user_data`中には連続する23ビット以上の"0"を配置することができないので、この禁止項目に触れないようにコンバータ212によりデータを変換するのである。

【0110】ユーザデータフォーマッタ213は、コンバータ212より供給される`converted_history_stream()`に、後述する図38に基づいて、`Data_ID`を付加し、さらに、`user_data_stream_code`を付加して、`video_str`

eam中に挿入できるuser_dataを生成し、符号化装置106に出力する。

【0111】図20は、ヒストリフォーマッタ211の構成例を表している。その符号語変換器301と符号長変換器305には、符号化パラメータ（今回、履歴情報として伝送する符号化パラメータ）（項目データ）と、この符号化パラメータを配置するストリームを特定する情報（例えば、シンタックスの名称）（例えば、後述するsequence_headerの名称）（項目NO.）が、符号化パラメータ分離装置105から供給されている。符号語変換器301は、入力された符号化パラメータを、指示されたシンタックスに対応する符号語に変換し、パレルシフタ302に出力する。パレルシフタ302は、符号語変換器301より入力された符号語を、アドレス発生回路306より供給されるシフト量に対応する分だけシフトし、バイト単位の符号語として、スイッチ303に出力する。アドレス発生回路306が出力するビットセレクト信号により切り換えられるスイッチ303は、ビット分設けられており、パレルシフタ302より供給される符号語を、RAM304に供給し、記憶させる。このときの書き込みアドレスは、アドレス発生回路306から指定される。また、アドレス発生回路306から読み出しアドレスが指定されたとき、RAM304に記憶されているデータ（符号語）が読み出され、後段のコンバータ212に供給されるとともに、必要に応じて、スイッチ303を介してRAM304に再び供給され、記憶される。

【0112】符号長変換器305は、入力されるシンタックスと符号化パラメータとから、その符号化パラメータの符号長を決定し、アドレス発生回路306に出力する。アドレス発生回路306は、入力された符号長に対応して、上述したシフト量、ビットセレクト信号、書き込みアドレス、または読み出しアドレスを生成し、それらを、それぞれパレルシフタ302、スイッチ303、またはRAM304に供給する。

【0113】以上のように、ヒストリフォーマッタ211は、いわゆる可変長符号化器として構成され、入力された符号化パラメータを可変長符号化して出力する。

【0114】図21は、以上のようにしてヒストリフォーマット化されたデータをデコードするヒストリデコーダ203の構成例を表している。このヒストリデコーダ203には、コンバータ202から供給された符号化パラメータのデータがRAM311に供給されて、記憶される。このときの書き込みアドレスは、アドレス発生回路315から供給される。アドレス発生回路315はまた、所定のタイミングで読み出しアドレスを発生し、RAM311に供給する。このとき、RAM311は、読み出しアドレスに記憶されているデータを読み出し、パレルシフタ312に出力する。パレルシフタ312は、アドレス発生回路315が出力するシフト量に対応する分だけ、入力されるデータをシフトし、逆符号長変換器31

3と逆符号語変換器314に出力する。

【0115】逆符号長変換器313にはまた、コンバータ202から、符号化パラメータが配置されているストリームのシンタックスの名称が供給されている。逆符号長変換器313は、そのシンタックスに基づいて、入力されたデータ（符号語）から符号長を求め、求めた符号長をアドレス発生回路315に出力する。

【0116】また、逆符号語変換器314は、パレルシフタ312より供給されたデータを、シンタックスに基づいて復号し（逆符号語化し）、符号化パラメータ多重装置103に出力する。

【0117】また、逆符号語変換器314は、どのような符号語が含まれているのかを特定するのに必要な情報（符号語の区切りを決定するのに必要な情報）を抽出し、アドレス発生回路315に出力する。アドレス発生回路315は、この情報と逆符号長変換器313より入力された符号長に基づいて、書き込みアドレスおよび読み出しアドレスを発生し、RAM311に出力するとともに、シフト量を発生し、パレルシフタ312に出力する。

【0118】図22は、コンバータ212の構成例を表している。この例においては、ヒストリフォーマッタ211とコンバータ212の間に配置されているバッファメモリ320の、コントローラ326が出力する読み出しアドレスから8ビットのデータが読み出され、D型フリップフロップ（D-FF）321に供給され、保持されるようになっている。そして、D型フリップフロップ321より読み出されたデータは、スタンプ回路323に供給されるとともに、8ビットのD型フリップフロップ322にも供給され、保持される。D型フリップフロップ322より読み出された8ビットのデータは、D型フリップフロップ321より読み出された8ビットのデータと合成され、16ビットのパラレルデータとして、スタンプ回路323に供給される。

【0119】スタンプ回路323は、コントローラ326より供給されるスタンプ位置を示す信号（stuff position）の位置に符号“1”を挿入し（スタンプイング）、合計17ビットのデータとして、パレルシフタ324に出力する。

【0120】パレルシフタ324は、コントローラ326より供給されるシフト量を示す信号（shift）に基づいて入力されたデータをシフトして、8ビットのデータを抽出し、8ビットのD型フリップフロップ325に出力する。D型フリップフロップ325に保持されたデータは、そこから読み出され、バッファメモリ327を介して、後段のユーザデータフォーマッタ213に供給される。この時、コントローラ326は、出力するデータとともに、書き込みアドレスを発生し、コンバータ212とユーザデータフォーマッタ213との間に介在するバッファメモリ327に供給する。

【0121】図23は、スタッフ回路323の構成例を表している。D型フリップフロップ322、321より入力された16ビットのデータは、それぞれスイッチ331-16乃至331-1の接点aに入力されている。スイッチ331-i (i=0乃至15)の接点cには、MSB側(図中上方)に隣接するスイッチのデータが供給されている。例えば、スイッチ331-12の接点cには、MSB側に隣接するスイッチ331-13の接点aに供給されているLSBから13番目のデータが供給されており、スイッチ331-13の接点cには、MSB側に隣接するスイッチ331-13の接点aに供給されているLSB側から14番目のデータが供給されている。

【0122】但し、LSBに対応するスイッチ331-1よりさらに下側のスイッチ331-0の接点aは、開放されている。また、MSBに対応するスイッチ331-16の接点cは、それより上位のスイッチが存在しないため、開放されている。

【0123】各スイッチ331-0乃至331-16の接点bには、データ"1"が供給されている。

【0124】デコーダ332は、コントローラ326より供給されるデータ"1"を挿入する位置を示す信号stuff positionに対応して、スイッチ331-0乃至331-16のうち、1つのスイッチを接点b側に切り替え、それよりLSB側のスイッチは、接点c側にそれぞれ切り替え、それよりMSB側のスイッチは、接点a側に切り替えさせる。

【0125】図23は、LSB側から13番目にデータ"1"を挿入する場合の例を示している。従って、この場合、スイッチ331-0乃至スイッチ331-12は、いずれも接点c側に切り替えられ、スイッチ331-13は、接点b側に切り替えられ、スイッチ331-14乃至スイッチ331-16は、接点a側に切り替えられている。

【0126】図22のコンバータ212は、以上のような構成により、22ビットの符号を23ビットに変換して、出力することになる。

【0127】図24は、図22のコンバータ212の各部の出力データのタイミングを表している。コンバータ212のコントローラ326がバイト単位のクロックに同期して、読み出しアドレス(図24(A))を発生すると、バッファメモリ320から、それに対応するデータが、バイト単位で読み出され、D型フリップフロップ321に一旦保持される。そして、D型フリップフロップ321より読み出されたデータ(図24(B))は、スタッフ回路323に供給されるとともに、D型フリップフロップ322に供給され、保持される。D型フリップフロップ322に保持されたデータは、そこからさらに読み出され(図24(C))、スタッフ回路323に供給される。

【0128】従って、スタッフ回路323の入力(図2

4(D))は、読み出しアドレスA1のタイミングにおいて、最初の1バイトのデータD0とされ、次の読み出しアドレスA2のタイミングにおいて、1バイトのデータD0と1バイトのデータD1より構成される2バイトのデータとなり、さらに読み出しアドレスA3のタイミングにおいては、データD1とデータD2より構成される2バイトのデータとなる。

【0129】スタッフ回路323には、データ"1"を挿入する位置を示す信号stuff position(図24

(E))がコントローラ326より供給される。スタッフ回路323のデコーダ332は、スイッチ331-16乃至331-0のうち、この信号stuff positionに対応するスイッチを接点bに切り換え、それよりLSB側のスイッチを接点c側に切り換え、さらにそれよりMSB側のスイッチを接点a側に切り換える。これにより、データ"1"が挿入されるので、スタッフ回路323からは、信号stuff positionで示す位置に、データ"1"が挿入されたデータ(図24(F))が出力される。

【0130】パレルシフタ324は、入力されたデータを、コントローラ326より供給される信号shift(図24(G))で示される量だけパレルシフトして、出力する(図24(H))。この出力がさらにD型フリップフロップ325で一旦保持された後、後段に出力される(図24(I))。

【0131】D型フリップフロップ325より出力されるデータには、22ビットのデータの次に、データ"1"が挿入されている。従って、データ"1"と、次のデータ"1"の間には、その間のビットが全て0であったとしても、0のデータの連続する数は22となる。

【0132】図25は、コンバータ202の構成例を表している。このコンバータ202のD型フリップフロップ341乃至コントローラ346よりなる構成は、図22に示したコンバータ212のD型フリップフロップ321乃至コントローラ326と基本的に同様の構成であるが、コンバータ212におけるスタッフ回路323に代えて、ディリット回路343が挿入されている点がコンバータ212における場合と異なっている。その他の構成は、図22のコンバータ212における場合と同様である。

【0133】すなわち、このコンバータ202においては、コントローラ346が出力する削除するビットの位置を示す信号delete positionに従って、ディリット回路343が、そのビット(図22のスタッフ回路323で挿入されたデータ"1")が削除される。

【0134】その他の動作は、図22のコンバータ212における場合と同様である。

【0135】図26は、ディリット回路343の構成例を表している。この構成例においては、D型フリップフロップ342、341より入力された16ビットのデータのうち、LSB側の15ビットが、それぞれ対応するス

スイッチ351-0乃至351-14の接点aに供給されている。各スイッチの接点bには、1ビットだけMSB側のデータが供給されている。デコーダ352は、コントローラ346より供給される信号delete positionにより指定されるビットを削除して、15ビットのデータとして出力するようになっている。

【0136】図26は、LSBから第13番目のビットがディリットされる状態を示している。従って、この場合、スイッチ351-0乃至スイッチ351-11が接点a側に切り替えられ、LSBから第12番目までの12ビットが、そのまま選択、出力されている。また、スイッチ351-12乃至351-14は、それぞれ接点b側に切り替えられているので、第14番目乃至第16番目のデータが、第13番目乃至第15番目のビットのデータとして選択、出力される。

【0137】図23のスタッフ回路323および図26のディリット回路343の入力が16ビットとなっているのは、それぞれ図22のコンバータ212のスタッフ回路323の入力が、D型フリップフロップ322、321より供給される16ビットとされており、また、図25のコンバータ202においても、ディリット回路343の入力が、D型フリップフロップ342、341により16ビットとされているためである。図22において、スタッフ回路323の出力する17ビットをバレルシフタ324でバレルシフトすることにより、例えば8ビットを最終的に選択、出力しているのと同様に、図25のコンバータ202においても、ディリット回路343の出力する15ビットのデータを、バレルシフタ344で所定量だけバレルシフトすることにより、8ビットのデータとしている。

【0138】図27は、コンバータ212の他の構成例を表している。この構成例においては、カウンタ361が入力データのうち、連続する0のビットの数をカウントし、そのカウント結果をコントローラ326に出力するようになっている。コントローラ326は、例えばカウンタ361が連続する0のビットを22個カウントしたとき、信号stuff positionをスタッフ回路323に出力する。また、このとき、コントローラ326は、カウンタ361をリセットし、再び連続する0のビットの数をカウンタ361にカウントさせる。

【0139】その他の構成と動作は、図22における場合と同様である。

【0140】図28は、コンバータ202の他の構成例を表している。この構成例においては、入力データのうち、連続する0の数をカウンタ371がカウントし、そのカウント結果をコントローラ346に出力するようになっている。カウンタ371のカウント値が22に達したとき、コントローラ346は、信号delete positionをディリット回路343に出力するとともに、カウンタ371をリセットし、再び新たな連続する0のビット

の数をカウンタ371にカウントさせる。その他の構成は、図25における場合と同様である。

【0141】このように、この構成例においては、所定のパターン（データ“0”の連続する数）に基づいて、マーカビットとしてのデータ“1”が挿入され、また、削除されることになる。

【0142】図27と図28に示す構成は、図22と図25に示す構成よりも効率的な処理が可能となる。但し、変換後の長さが元の履歴情報に依存することになる。

【0143】図29は、ユーザデータフォーマット213の構成例を表している。この例においては、コントローラ383がコンバータ212とユーザデータフォーマット213との間に配置されているバッファメモリ（図示せず）に読み出しアドレスを出力すると、そこから読み出されたデータが、ユーザデータフォーマット213のスイッチ382の接点a側に供給される。ROM381には、ユーザデータスタートコード、データIDなどのuser_data()を生成するのに必要なデータが記憶されている。コントローラ313は、所定のタイミングにおいて、スイッチ382を接点a側または接点b側に切り替え、ROM381に記憶されているデータ、またはコンバータ212より供給されるデータを適宜選択し、出力する。これにより、user_data()のフォーマットのデータが符号化装置106に出力される。

【0144】なお、図示は省略するが、ユーザデータコーダ201は、図29のROM381より読み出され、挿入されたデータを削除するスイッチを介して、入力データを出力するようにすることで実現することができる。

【0145】図30は、例えば映像編集スタジオにおいて、複数のトランスコーダ101-1乃至101-Nが直列に接続されて使用される状態を示している。各トランスコーダ101-i（i=1乃至N）の符号化パラメータ多重装置103-iは、上述した符号化パラメータ用の領域の最も古い符号化パラメータが記録されている区画に、自己が用いた最新の符号化パラメータを上書きする。このことにより、ベースバンドの画像データには、同一のマクロブロックに対応する直近の4世代分の符号化パラメータ（世代履歴情報）が記録されることになる。

【0146】各符号化装置106-iのエンコーダ121-i（図19）は、その可変長符号化回路58において、符号化パラメータ分離装置105-iから供給される今回用いる符号化パラメータに基づいて、量子化回路57より供給されるビデオデータを符号化する。このようにして生成されるビットストリーム（例えば、picture_header()中に、その現符号化パラメータは多重化される。

【0147】可変長符号化回路58はまた、履歴符号化

装置107-iより供給されるユーザデータ（世代履歴情報を含む）を、出力するビットストリーム中に多重化する（図18に示すような埋め込み処理ではなく、ビットストリーム中に多重化される）。そして、符号化装置106-iの出力するビットストリームは、SDT1108-iを介して、後段のトランスコーダ101-(i+1)に入力される。

【0148】トランスコーダ101-iとトランスコーダ101-(i+1)は、それぞれ図15に示すように構成されている。従って、その処理は、図15を参照して説明した場合と同様となる。

【0149】実際の符号化パラメータの履歴を利用した符号化として、現在Iピクチャとして符号化されていたものを、PもしくはBピクチャに変更したい場合、過去の符号化パラメータの履歴を見て、過去にPもしくはBピクチャであった場合を探し、これらの履歴が存在した場合は、その動きベクトルなどのパラメータを利用して、ピクチャタイプを変更する。反対に過去に履歴がない場合は、動き検出を行わないピクチャタイプの変更を断念する。もちろん履歴がない場合であっても、動き検出を行えばピクチャタイプを変更できる。

【0150】図18に示すフォーマットの場合、4世代分の符号化パラメータを埋め込むようにしたが、I、P、Bの各ピクチャタイプのパラメータを埋め込むようにすることもできる。図31は、この場合のフォーマットの例を示している。この例では、同一のマクロブロックが、過去にピクチャタイプの変更を伴って符号化されたときにおける、ピクチャタイプ毎に1世代分の符号化パラメータ（ピクチャ履歴情報）が記録される。したがって、図16に示したデコーダ111、および図19に示したエンコーダ121は、現在（最新）、1世代前、2世代前、および3世代前の符号化パラメータの代わりに、Iピクチャ、Pピクチャ、およびBピクチャに対応する1世代分の符号化パラメータを入出力することになる。

【0151】また、この例の場合、Cb[1][x]とCr[1][x]の領域は利用しないので、Cb[1][x]とCr[1][x]の領域を有さない4:2:0フォーマットの画像データにも本発明を適用することができる。

【0152】この例の場合、復号装置102は、符号化パラメータを復号と同時に取り出し、ピクチャタイプを判定して、画像信号のピクチャタイプに対応した場所に符号化パラメータを書き込んで（多重化して）符号化パラメータ分離装置105に出力する。符号化パラメータ分離装置105は、符号化パラメータを分離し、これから符号化したいピクチャタイプと、入力された過去の符号化パラメータを考慮して、ピクチャタイプを変更しながら再符号化を行うことができる。

【0153】次に、各トランスコーダ101において、変更が可能なピクチャタイプを判定する処理について、

図32のフローチャートを参照して説明する。なお、この処理はトランスコーダ101におけるピクチャタイプの変更は、過去の動きベクトルを利用するので、動き検出を行わないで実行されることを前提としている。また、以下に説明する処理は、符号化パラメータ分離装置105により実行される。

【0154】ステップS1において、ピクチャタイプ毎に1世代分の符号化パラメータ（ピクチャ履歴情報）が符号化パラメータコントローラ122に入力される。

【0155】ステップS2において、符号化パラメータ分離装置105は、ピクチャ履歴情報にBピクチャに変更したときの符号化パラメータが存在するか否かを判定する。ピクチャ履歴情報にBピクチャに変更したときの符号化パラメータが存在すると判定された場合、ステップS3に進む。

【0156】ステップS3において、符号化パラメータ分離装置105は、ピクチャ履歴情報にPピクチャに変更したときの符号化パラメータが存在するか否かを判定する。ピクチャ履歴情報にPピクチャに変更したときの符号化パラメータが存在すると判定された場合、ステップS4に進む。

【0157】ステップS4において、符号化パラメータ分離装置105は、変更可能なピクチャタイプがIピクチャ、Pピクチャ、およびBピクチャであると判断する。

【0158】ステップS3において、ピクチャ履歴情報にPピクチャに変更したときの符号化パラメータが存在しないと判定された場合、ステップS5に進む。

【0159】ステップS5において、符号化パラメータ分離装置105は、変更可能なピクチャタイプがIピクチャ、およびBピクチャであると判断する。さらに、符号化パラメータ分離装置105は、特殊処理（Bピクチャの履歴情報に含まれる後方予測ベクトルを使わず、前方予測ベクトルだけを使う）を施すことにより、擬似的にPピクチャに変更可能であると判断する。

【0160】ステップS2において、ピクチャ履歴情報にBピクチャに変更したときの符号化パラメータが存在しないと判定された場合、ステップS6に進む。

【0161】ステップS6において、符号化パラメータ分離装置105は、ピクチャ履歴情報にPピクチャに変更したときの符号化パラメータが存在するか否かを判定する。ピクチャ履歴情報にPピクチャに変更したときの符号化パラメータが存在すると判定された場合、ステップS7に進む。

【0162】ステップS7において、符号化パラメータ分離装置105は、変更可能なピクチャタイプがIピクチャ、およびPピクチャであると判断する。さらに、符号化パラメータ分離装置105は、特殊処理（Pピクチャに履歴情報に含まれる前方予測ベクトルだけを使う）を施すことにより、Bピクチャに変更可能であると判断

する。

【0163】ステップS6において、ピクチャ履歴情報にPピクチャに変更したときの符号化パラメータが存在しないと判定された場合、ステップS8に進む。ステップS8において、符号化パラメータ分離装置105は、動きベクトルが存在しないので、変更可能なピクチャタイプがIピクチャだけである（IピクチャなのでIピクチャ以外には変更できない）と判断する。

【0164】ステップS4、S5、S7、S8の処理の次にステップS9において、符号化パラメータ分離装置105は、変更可能なピクチャタイプを表示装置（図示せず）に表示してユーザに通知する。

【0165】図33は、ピクチャタイプ変更の例を示している。ピクチャタイプの変更は、GOPを構成するフレーム数が変更される。すなわち、この例の場合、 $N=15$ （GOPのフレーム数 $N=15$ ）、 $M=3$ （GOP内のI、またはPピクチャの出現周期 $M=3$ ）のフレームから構成されるLong GOP（第1世代）から、 $N=1$ 、 $M=1$ のフレームで構成されるShort GOP（第2世代）に変換され、再度、 $N=15$ 、 $M=3$ のフレームから構成されるLong GOP（第3世代）に変換されている。なお、図中において破線は、GOPの境界を示している。

【0166】第1世代から第2世代にピクチャタイプが変更される場合において、上述した変更可能ピクチャタイプ判定処理の説明から明らかなように、全てのフレームは、ピクチャタイプをIピクチャに変更することが可能である。このピクチャタイプ変更のとき、動画像（第0世代）が第1世代に変換されたときに演算された全ての動きベクトルは、ピクチャ履歴情報に保存された（残された）状態となる。次に、再度Long GOPに変換される（第2世代から第3世代にピクチャタイプが変更される）場合、第0世代から第1世代に変換されたときのピクチャタイプ毎の動きベクトルが保存されているので、これを再利用することにより、画質劣化を抑えて、再度、Long GOPに変換することが可能となる。

【0167】図34は、ピクチャタイプ変更の他の例を示している。この例の場合、 $N=14$ 、 $M=2$ であるLong GOP（第1世代）から、 $N=2$ 、 $M=2$ であるShort GOP（第2世代）に変換され、さらに、 $N=1$ 、 $M=1$ であるフレーム数が1のShort GOP（第3世代）に変換されて、フレーム数 N がランダムなGOP（第4世代）に変換される。

【0168】この例においても、第0世代から第1世代に変換されたときのピクチャタイプ毎の動きベクトルが、第3世代から第4世代への変換のときまで保存される。そこで、図34に示すように、複雑にピクチャタイプを変更しても、保存されている符号化パラメータを再利用されることにより、画質劣化を小さく抑えることができる。さらに、保存されている符号化パラメータの量子化スケールを有効に利用すれば画質劣化の少ない符号化を実現できる。

【0169】この量子化スケールの再利用について、図35を参照して説明する。図35は、所定のフレームが、第1世代から第4世代まで常に、Iピクチャに変換されており、ビットレートだけが、4Mbps、18Mbps、または50Mbpsに変更されていることを示している。

【0170】例えば、第1世代（4Mbps）から第2世代（18Mbps）への変換の際に、ビットレートの高速化に伴って、細かい量子化スケールで再符号化しても画質は向上しない。なぜならば、過去において粗い量子化ステップで量子化されたデータは、復元しないからである。したがって、図35に示すように、途中でビットレートが高速化しても、それに伴って細かい量子化ステップで量子化することは、情報量が増加するだけであって画質の向上には繋がらない。したがって、過去のもっとも粗い（大きい）量子化スケールを維持するように制御すれば、最も無駄が無く、効率的な符号化が可能となる。

【0171】上述したように、ビットレートが変更されるときは、過去の量子化スケールの履歴を利用して符号化することは非常に有効である。

【0172】この量子化制御処理について、図36のフローチャートを参照して説明する。ステップS11において、符号化パラメータ分離装置105は、入力されたピクチャ履歴情報に、いまから変換するピクチャタイプの符号化パラメータが存在するか否かを判定する。変換するピクチャタイプの符号化パラメータが存在すると判定された場合、ステップS12に進む。

【0173】ステップS12において、符号化パラメータ分離装置105は、ピクチャ履歴情報の対照となる符号化パラメータから量子化スケール（ $Q_history$ ）を抽出する。

【0174】ステップS13において、符号化パラメータ分離装置105は、送信バッファ59から量子化回路57にフィードバックされる量子化スケールの候補値 $Q_feedback$ を読み取る。

【0175】ステップS14において、符号化パラメータ分離装置105は、 $Q_history$ が $Q_feedback$ よりも大きい（粗い）か否かを判定する。 $Q_history$ が $Q_feedback$ よりも大きいと判定された場合、ステップS15に進む。

【0176】ステップS15において、符号化パラメータ分離装置105は、量子化スケールとして $Q_history$ を量子化回路57に出力する。量子化回路57は、 $Q_history$ を用いて量子化を実行する。

【0177】ステップS16において、フレームに含まれる全てのマクロブロックが量子化されたか否かが判定される。全てのマクロブロックが量子化されていないと判定された場合、ステップS13に戻り、ステップS13乃至S16の処理が、全てのマクロブロックが量子化されるまで繰り返される。

【0178】ステップS14において、 $Q_history$ が Q_f

eedbackよりも大きくない（細かい）いと判定された場合、ステップS 17に進む。

【0179】ステップS 17において、符号化パラメータ分離装置105は、量子化スケールとしてQ_feedbackを量子化回路57に出力する。量子化回路57は、Q_feedbackを用いて量子化を実行する。

【0180】ステップS 11において、変換するピクチャタイプの符号化パラメータが存在しないと判定された場合、ステップS 18に進む。

【0181】ステップS 18において、量子化回路57は、送信バッファ59からフィードバックされる量子化スケールの候補値Q_feedbackを受け付ける。

【0182】ステップS 19において、量子化回路57は、Q_feedbackを用いて量子化を実行する。

【0183】ステップS 20において、フレームに含まれる全てのマクロブロックが量子化されたか否かが判定される。全てのマクロブロックが量子化されていないと判定された場合、ステップS 18に戻り、ステップS 18乃至S 20の処理が、全てのマクロブロックが量子化されるまで繰り返される。

【0184】なお、本実施の形態におけるトランスコーダ101の内部においては、上述したように、復号側と符号側が粗結合されており、符号化パラメータを画像データに多重化させて伝送させたが、図37に示すように、復号装置102と符号化装置106を符号化パラメータ伝送用の高速バス111で接続する（密結合する）ようにしてもよい。

【0185】図38は、MPEGのビデオストリームをデコードするためのシンタックスを表わした図である。デコーダは、このシンタックスに従ってMPEGビットストリームをデコードすることによって、ビットストリームから意味のある複数のデータ項目（データエレメント）を抽出する。以下に説明するシンタックスは、図において、その関数や条件文は細活字で表わされ、そのデータエレメントは、太活字で表されている。データ項目は、その名称、ビット長、及びそのタイプと伝送順序を示すニーモニック（Mnemonic）で記述されている。

【0186】まず、この図38に示されているシンタックスにおいて使用されている関数について説明する。

【0187】next_start_code()関数は、ビットストリーム中に記述されているスタートコードを探すための関数である。よって、この図38に示されたシンタックスにおいて、このnext_start_code()関数の次に、sequence_header()関数とsequence_extension()関数とが順に配置されているので、このビットストリームには、このsequence_header()関数とsequence_extension()関数によって定義されたデータエレメントが記述されている。従って、ビットストリームのデコード時には、このnext_start_code()関数によって、sequence_header()関数とsequence_extension()関数の先頭に記述されているスター

トコード（データエレメントの一種）をビットストリーム中から見つけ、それを基準にして、sequence_header()関数とsequence_extension()関数をさらに見つけ、それらによって定義された各データエレメントをデコードする。

【0188】尚、sequence_header()関数は、MPEGビットストリームのシーケンス層のヘッダデータを定義するための関数であって、sequence_extension()関数は、MPEGビットストリームのシーケンス層の拡張データを定義するための関数である。

【0189】sequence_extension()関数の次に配置されているdo{}while構文は、while文によって定義されている条件が真である間、do文の{}内の関数に基いて記述されたデータエレメントをデータストリーム中から抽出するための構文である。すなわち、do{}while構文によって、while文によって定義されている条件が真である間、ビットストリーム中から、do文内の関数に基いて記述されたデータエレメントを抽出するデコード処理が行われる。

【0190】このwhile文に使用されているnextbits()関数は、ビットストリーム中に現れるビット又はビット列と、次にデコードされるデータエレメントとを比較するための関数である。この図38のシンタックスの例では、nextbits()関数は、ビットストリーム中のビット列とビデオシーケンスの終わりを示すsequence_end_codeとを比較し、ビットストリーム中のビット列とsequence_end_codeとが一致しないときに、このwhile文の条件が真となる。従って、sequence_extension()関数の次に配置されているdo{}while構文は、ビットストリーム中に、ビデオシーケンスの終わりを示すsequence_end_codeが現れない間、do文中の関数によって定義されたデータエレメントがビットストリーム中に記述されていることを示している。

【0191】ビットストリーム中には、sequence_extension()関数によって定義された各データエレメントの次には、extension_and_user_data(0)関数によって定義されたデータエレメントが記述されている。このextension_and_user_data(0)関数は、MPEGビットストリームのシーケンス層の拡張データとユーザデータを定義するための関数である。

【0192】このextension_and_user_data(0)関数の次に配置されているdo{}while構文は、while文によって定義されている条件が真である間、do文の{}内の関数に基いて記述されたデータエレメントを、ビットストリーム中から抽出するための関数である。このwhile文において使用されているnextbits()関数は、ビットストリーム中に現れるビット又はビット列と、picture_start_code又はgroup_start_codeとの一致を判断するための関数であって、ビットストリーム中に現れるビット又はビット列と、picture_start_code又はgroup_start_codeと

が一致する場合には、while文によって定義された条件が真となる。よって、このdo{ }while構文は、ビットストリーム中において、picture_start_code又はgroup_start_codeが現れた場合には、そのスタートコードの次に、do文中の関数によって定義されたデータエレメントのコードが記述されているので、このpicture_start_code又はgroup_start_codeによって示されるスタートコードを探し出すことによって、ビットストリーム中からdo文中に定義されたデータエレメントを抽出することができる。

【0193】このdo文の最初に記述されているif文は、ビットストリーム中にgroup_start_codeが現れた場合、という条件を示している。このif文による条件が真である場合には、ビットストリーム中には、このgroup_start_codeの次にgroup_of_picture_header(1)関数及びextension_and_user_data(1)関数によって定義されているデータエレメントが順に記述されている。

【0194】このgroup_of_picture_header(1)関数は、MPEGビットストリームのGOP層のヘッダデータを定義するための関数であって、extension_and_user_data(1)関数は、MPEGビットストリームのGOP層の拡張データ(extension_data)及びユーザデータ(user_data)を定義するための関数である。

【0195】さらに、このビットストリーム中には、group_of_picture_header(1)関数及びextension_and_user_data(1)関数によって定義されているデータエレメントの次に、picture_header()関数とpicture_coding_extension()関数によって定義されたデータエレメントが記述されている。もちろん、先に説明したif文の条件が真とならない場合には、group_of_picture_header(1)関数及びextension_and_user_data(1)関数によって定義されているデータエレメントは記述されていないので、extension_and_user_data(0)関数によって定義されているデータエレメントの次に、picture_header()関数とpicture_coding_extension()関数によって定義されたデータエレメントが記述されている。

【0196】このpicture_header()関数は、MPEGビットストリームのピクチャ層のヘッダデータを定義するための関数であって、picture_coding_extension()関数は、MPEGビットストリームのピクチャ層の第1の拡張データを定義するための関数である。

【0197】次のwhile文は、このwhile文によって定義されている条件が真である間、次のif文の条件判断を行うための関数である。このwhile文において使用されているnextbits()関数は、ビットストリーム中に現れるビット列と、extension_start_code又はuser_data_start_codeとの一致を判断するための関数であって、ビットストリーム中に現れるビット列と、extension_start_code又はuser_data_start_codeとが一致する場合には、このwhile文によって定義された条件が真となる。

【0198】第1のif文は、ビットストリーム中に現れるビット列とextension_start_codeとの一致を判断するための関数である。ビットストリーム中に現れるビット列と32ビットのextension_start_codeとが一致する場合には、ビットストリーム中において、extension_start_codeの次にextension_data(2)関数によって定義されるデータエレメントが記述されていることを示している。

【0199】第2のif文は、ビットストリーム中に現れるビット列とuser_data_start_codeとの一致を判断するための関数であって、ビットストリーム中に現れるビット列と32ビットのuser_data_start_codeとが一致する場合には、第3のif文の条件判断が行われる。このuser_data_start_codeは、MPEGビットストリームのピクチャ層のユーザデータエリアの開始を示すためのスタートコードである。

【0200】第3のif文は、ビットストリーム中に現れるビット列とHistory_Data_IDとの一致を判断するための関数である。ビットストリーム中に現れるビット列とこの8ビットのHistory_Data_IDとが一致する場合には、このMPEGビットストリームのピクチャ層のユーザデータエリアにおいて、この8ビットのHistory_Data_IDによって示されるコードの次に、converted_history_stream()関数によって定義されるデータエレメントが記述されていることを示している。

【0201】converted_history_stream()関数は、MPEG符号化時に使用したあらゆる符号化パラメータを送送するための履歴情報及び履歴データを記述するための関数である。このconverted_history_stream()関数によって定義されているデータエレメントの詳細は後述する。また、このHistory_Data_IDは、MPEGビットストリームのピクチャ層のユーザデータエリアに記述されたこの履歴情報及び履歴データが記述されている先頭を示すためのスタートコードである。

【0202】else文は、第3のif文において、条件が非真であることを示すための関数である。従って、このMPEGビットストリームのピクチャ層のユーザデータエリアにおいて、converted_history_stream()関数によって定義されたデータエレメントが記述されていない場合には、user_data()関数によって定義されたデータエレメントが記述されている。

【0203】picture_data()関数は、MPEGビットストリームのピクチャ層のユーザデータの次に、スライス層及びマクロブロック層に関するデータエレメントを記述するための関数である。通常は、このpicture_data()関数によって示されるデータエレメントは、ビットストリームのピクチャ層のユーザデータエリアに記述されたconverted_history_stream()関数によって定義されるデータエレメント又はuser_data()関数によって定義されたデータエレメントの次に記述されているが、ピクチャ

層のデータエレメントを示すビットストリーム中に、`extension_start_code`又は`user_data_start_code`が存在しない場合には、この`picture_data()`関数によって示されるデータエレメントは、`picture_coding_extension()`関数によって定義されるデータエレメントの次に記述されている。

【0204】この`picture_data()`関数によって示されるデータエレメントの次には、`sequence_header()`関数と`sequence_extension()`関数とによって定義されたデータエレメントが順に配置されている。この`sequence_header()`関数と`sequence_extension()`関数によって記述されたデータエレメントは、ビデオストリームのシーケンスの先頭に記述された`sequence_header()`関数と`sequence_extension()`関数によって記述されたデータエレメントと全く同じである。このように同じデータをストリーム中に記述する理由は、ビットストリーム受信装置側でデータストリームの途中（例えばピクチャ層に対応するビットストリーム部分）から受信が開始された場合に、シーケンス層のデータを受信できなくなり、ストリームをデコード出来なくなることを防止するためである。

【0205】この最後の`sequence_header()`関数と`sequence_extension()`関数とによって定義されたデータエレメントの次、つまり、データストリームの最後には、シーケンスの終わりを示す32ビットの`sequence_end_code`が記述されている。

【0206】以上のシンタックスの基本的な構成の概略を示すと、図39に示すようになる。

【0207】次に、`converted_history_stream()`関数によって定義されたヒストリーストリームに関して説明する。

【0208】この`converted_history_stream()`は、MGEのピクチャ層のユーザデータエリアに履歴情報を示すヒストリーストリームを挿入するための関数である。尚、「converted」の意味は、スタートエミュレーションを防止するために、ユーザエリアに挿入すべき履歴データから構成される履歴ストリームの少なくとも22ビット毎にマーカービット（1ビット）を挿入する変換処理を行ったストリームであることを意味している。

【0209】この`converted_history_stream()`は、以下に説明する固定長の履歴ストリーム（図40乃至図46）又は可変長の履歴ストリーム（図47）のいずれかの形式で記述される。エンコーダ側において固定長の履歴ストリームを選択した場合には、デコーダ側において履歴ストリームから各データエレメントをデコードするための回路及びソフトウェアが簡単になるというメリットがある。一方、エンコーダ側において可変長の履歴ストリームを選択した場合には、エンコーダにおいてピクチャ層のユーザエリアに記述される履歴情報（データエレメント）を必要に応じて任意に選択することができるので、履歴ストリームのデータ量を少なくすることがで

き、その結果、符号化されたビットストリーム全体のデータレートを低減することができる。

【0210】本発明において説明する「履歴情報」「履歴データ」「履歴パラメータ」とは、過去の符号化処理において使用した符号化パラメータ（又はデータエレメント）のことであって、現在の（最終段の）符号化処理において使用した符号化パラメータのことではない。例えば、第1世代の符号化処理において、あるピクチャを1ピクチャで符号化して伝送し、次なる第2世代の符号化処理において、このピクチャを今度はPピクチャとして符号化して伝送し、さらに、第3世代の符号化処理において、このピクチャをBピクチャで符号化して伝送する例をあげて説明する。第3世代の符号化処理において使用した符号化パラメータが、第3世代の符号化処理において生成された符号化ビットストリームのシーケンス層、GOP層、ピクチャ層、スライス層及びマクロブロック層の所定位置に記述されている。一方、過去の符号化処理である第1世代及び第2世代の符号化処理において使用した符号化パラメータは、第3世代の符号化処理において使用した符号化パラメータが記述されるシーケンス層やGOP層に記述されるのでは無く、既に説明したシンタックスに従って、符号化パラメータの履歴情報として、ピクチャ層のユーザデータエリアに記述される。

【0211】まず、固定長の履歴ストリームシンタックスについて図40乃至図46を参照して説明する。

【0212】最終段（例えば第3世代）の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、まず最初に、過去（例えば第1世代及び第2世代）の符号化処理において使用されていたシーケンス層のシーケンスヘッダに関する符号化パラメータが、履歴ストリームとして挿入される。尚、過去の符号化処理において生成されたビットストリームのシーケンス層のシーケンスヘッダ等の履歴情報は、最終段の符号化処理において生成されたビットストリームのシーケンス層のシーケンスヘッダに挿入されることは無いという点に注意すべきである。

【0213】過去の符号化処理で使用したシーケンスヘッダに関するデータエレメントは、`sequence_header_code`、`sequence_header_present_flag`、`horizontal_size_value`、`vertical_size_value`、`aspect_ratio_information`、`frame_rate_code`、`bit_rate_value`、`marker_bit`、`VBV_buffer_size_value`、`constrained_parameter_flag`、`load_intra_quantizer_matrix`、`intra_quantizer_matrix`、`load_non_intra_quantizer_matrix`、及び`non_intra_quantizer_matrix`等から構成される。

【0214】`sequence_header_code`は、シーケンス層のスタート同期コードを表すデータである。`sequence_header_present_flag`は、`sequence_header`内のデータが有効か無効かを示すデータである。`horizontal_size_val`

uelは、画像の水平方向の画素数の下位12ビットから成るデータである。vertical_size_valueは、画像の縦のライン数の下位12ビットからなるデータである。aspect_ratio_informationは、画素のアスペクト比（縦横比）または表示画面アスペクト比を表すデータである。frame_rate_codeは、画像の表示周期を表すデータである。

【0215】bit_rate_valueは、発生ビット量に対する制限のためのビット・レートの下位18ビット(400bsp単位で切り上げる)データである。marker_bitは、スタートコードエミュレーションを防止するために挿入されるビットデータである。Vbv_buffer_size_valueは、発生符号量制御用の仮想バッファ（ビデオバッファペリファイヤー）の大きさを決める値の下位10ビットデータである。constrained_parameter_flagは、各パラメータが制限以内であることを示すデータである。load_intra_quantizer_matrixは、イントラMB用量子化マトリックス・データの存在を示すデータである。intra_quantizer_matrixは、イントラMB用量子化マトリックスの値を示すデータである。load_non_intra_quantizer_matrixは、非イントラMB用量子化マトリックス・データの存在を示すデータである。non_intra_quantizer_matrixは、非イントラMB用量子化マトリックスの値を表すデータである。

【0216】次に、最終段の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたシーケンス層のシーケンスエクステンションを表わすデータエレメントが、履歴ストリームとして記述される。

【0217】この過去の符号化処理で使用したシーケンスエクステンションを表わすデータエレメントは、extension_start_code、extension_start_code_identifier、sequence_extension_present_flag、profile_and_level_indication、progressive_sequence、chroma_format、horizontal_size_extension、vertical_size_extension、bit_rate_extension、vbv_buffer_size_extension、low_delay、frame_rate_extension_n、及び frame_rate_extension_d等のデータエレメントである。

【0218】extension_start_codeは、エクステンションデータのスタート同期コードを表すデータである。extension_start_code_identifierは、どの拡張データが送られるかを示すデータである。sequence_extension_present_flagは、シーケンスエクステンション内のデータが有効であるか無効であるかを示すデータである。profile_and_level_indicationは、ビデオデータのプロファイルとレベルを指定するためのデータである。progressive_sequenceは、ビデオデータが順次走査であることを示すデータである。chroma_formatは、ビデオデータの色差フォーマットを指定するためのデータである。

【0219】horizontal_size_extensionは、シーケンスヘッダのhorizontal_size_valueに加える上位2ビット

のデータである。vertical_size_extensionは、シーケンスヘッダのvertical_size_valueに加える上位2ビットのデータである。bit_rate_extensionは、シーケンスヘッダのbit_rate_valueに加える上位12ビットのデータである。vbv_buffer_size_extensionは、シーケンスヘッダのvbv_buffer_size_valueに加える上位8ビットのデータである。low_delayは、Bピクチャを含まないことを示すデータである。frame_rate_extension_nは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。frame_rate_extension_dは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。

【0220】続いて、ビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたシーケンス層のシーケンスディスプレイエクステンションを表わすデータエレメントが、履歴ストリームとして記述される。

【0221】このシーケンスディスプレイエクステンションとして記述されているデータエレメントは、extension_start_code、extension_start_code_identifier、sequence_display_extension_present_flag、video_format、color_description、color_primaries、transfer_characteristics、matrix_coefficients、display_horizontal_size、及びdisplay_vertical_sizeから構成される。

【0222】extension_start_codeは、エクステンションデータのスタート同期コードを表すデータである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。sequence_display_extension_present_flagは、シーケンスディスプレイエクステンション内のデータエレメントが有効か無効かを示すデータである。video_formatは、原信号の映像フォーマットを表すデータである。color_descriptionは、色空間の詳細データがあることを示すデータである。color_primariesは、原信号の色特性の詳細を示すデータである。transfer_characteristicsは、光電変換がどのように行われたのかの詳細を示すデータである。matrix_coefficientsは、原信号が光の三原色からどのように変換されたかの詳細を示すデータである。display_horizontal_sizeは、意図するディスプレイの活性領域（水平サイズ）を表すデータである。display_vertical_sizeは、意図するディスプレイの活性領域（垂直サイズ）を表すデータである。

【0223】続いて、最終段の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において生成されたマクロブロックの位相情報を示すマクロブロックアサインメントデータ（macroblock_assignment_in_user_data）が、履歴ストリームとして記述される。

【0224】このマクロブロックの位相情報を示すmacroblock_assignment_in_user_dataは、macroblock_assignment_present_flag、v_phase、h_phase等のデータエレメントから構成される。

【0225】このmacroblock_assignment_present_flagは、macroblock_assignment_in_user_data内のデータエレメントが有効か無効かを示すデータである。v_phaseは、画像データからマクロブロックを切り出す際の垂直方向の位相情報を示すデータである。h_phaseは、画像データからマクロブロックを切り出す際の水平方向の位相情報を示すデータである。

【0226】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたGOP層のGOPヘッダを表わすデータエレメントが、履歴ストリームとして記述されている。

【0227】このGOPヘッダを表わすデータエレメントは、group_start_code、group_of_picture_header_present_flag、time_code、closed_gop、及びbroken_linkから構成される。

【0228】group_start_codeは、GOP層の開始同期コードを示すデータである。group_of_picture_header_present_flagは、group_of_picture_header内のデータエレメントが有効であるか無効であるかを示すデータである。time_codeは、GOPの先頭ピクチャのシーケンスの先頭からの時間を示すタイムコードである。closed_gopは、GOP内の画像が他のGOPから独立再生可能なことを示すフラグデータである。broken_linkは、編集などのためにGOP内の先頭のBピクチャが正確に再生できないことを示すフラグデータである。

【0229】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたピクチャ層のピクチャヘッダを表わすデータエレメントが、履歴ストリームとして記述されている。

【0230】このピクチャヘッダに関するデータエレメントは、picture_start_code、temporal_reference、picture_coding_type、vbv_delay、full_pel_forward_vector、forward_f_code、full_pel_backward_vector、及びbackward_f_codeから構成される。

【0231】具体的には、picture_start_codeは、ピクチャ層の開始同期コードを表すデータである。temporal_referenceは、ピクチャの表示順を示す番号でGOPの先頭でリセットされるデータである。picture_coding_typeは、ピクチャタイプを示すデータである。vbv_delayは、ランダムアクセス時の仮想バッファの初期状態を示すデータである。full_pel_forward_vectorは、順方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。forward_f_codeは、順方向動きベクトル探索範囲を表すデータである。full_pel_backward_vector

は、逆方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。backward_f_codeは、逆方向動きベクトル探索範囲を表すデータである。

【0232】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたピクチャ層のピクチャコーディングエクステンションが、履歴ストリームとして記述されている。

【0233】このピクチャコーディングエクステンションに関するデータエレメントは、extension_start_code、extension_start_code_identifier、f_code[0][0]、f_code[0][1]、f_code[1][0]、f_code[1][1]、intra_dc_precision、picture_structure、top_field_first、frame_predictive_frame_dct、concealment_motion_vectors、q_scale_type、intra_vlc_format、alternate_scan、repeat_firt_field、chroma_420_type、progressive_frame、composite_display_flag、v_axis、field_sequence、sub_carrier、burst_amplitude、及びsub_carrier_phaseから構成される。

【0234】extension_start_codeは、ピクチャ層のエクステンションデータのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。f_code[0][0]は、フォワード方向の水平動きベクトル探索範囲を表すデータである。f_code[0][1]は、フォワード方向の垂直動きベクトル探索範囲を表すデータである。f_code[1][0]は、バックワード方向の水平動きベクトル探索範囲を表すデータである。f_code[1][1]は、バックワード方向の垂直動きベクトル探索範囲を表すデータである。

【0235】intra_dc_precisionは、DC係数の精度を表すデータである。picture_structureは、フレームストラクチャかフィールドストラクチャかを示すデータである。フィールドストラクチャの場合は、上位フィールドか下位フィールドかもあわせて示すデータである。top_field_firstは、フレームストラクチャの場合、最初のフィールドが上位か下位かを示すデータである。frame_predictive_frame_dctは、フレーム・ストラクチャの場合、フレーム・モードDCTの予測がフレーム・モードだけであることを示すデータである。concealment_motion_vectorsは、イントラマクロブロックに伝送エラーを隠蔽するための動きベクトルがついていることを示すデータである。

【0236】q_scale_typeは、線形量子化スケールを利用するか、非線形量子化スケールを利用するかを示すデータである。intra_vlc_formatは、イントラマクロブロックに、別の2次元VLCを使うかどうかを示すデータである。alternate_scanは、ジグザグスキャンを使うか、オルタネート・スキャンを使うかの選択を表すデータである。repeat_firt_fieldは、2:3プルダウンの際に使われるデータである。chroma_420_typeは、信号

フォーマットが4:2:0の場合、次のprogressive_frameと同じ値、そうでない場合は0を表すデータである。progressive_frameは、このピクチャが、順次走査できているかどうかを示すデータである。composite_display_flagは、ソース信号がコンポジット信号であったかどうかを示すデータである。

【0237】v_axisは、ソース信号が、PALの場合に使われるデータである。field_sequenceは、ソース信号が、PALの場合に使われるデータである。sub_carrierは、ソース信号が、PALの場合に使われるデータである。burst_amplitudeは、ソース信号が、PALの場合に使われるデータである。sub_carrier_phaseは、ソース信号が、PALの場合に使われるデータである。

【0238】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用された量子化マトリックスエクステンションが、履歴ストリームとして記述されている。

【0239】この量子化マトリックスエクステンションに関するデータエレメントは、extension_start_code、extension_start_code_identifier、quant_matrix_extension_present_flag、load_intra_quantizer_matrix、intra_quantizer_matrix[64]、load_non_intra_quantizer_matrix、non_intra_quantizer_matrix[64]、load_chroma_intra_quantizer_matrix、chroma_intra_quantizer_matrix[64]、load_chroma_non_intra_quantizer_matrix、及びchroma_non_intra_quantizer_matrix[64]から構成される。

【0240】extension_start_codeは、この量子化マトリックスエクステンションのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。quant_matrix_extension_present_flagは、この量子化マトリックスエクステンション内のデータエレメントが有効か無効かを示すためのデータである。load_intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【0241】load_non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスの値を表すデータである。load_chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックス・データの存在を示すデータである。chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックスの値を示すデータである。load_chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックス・デ

ータの存在を示すデータである。chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【0242】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたコピーライトエクステンションが、履歴ストリームとして記述されている。

【0243】このコピーライトエクステンションに関するデータエレメントは、extension_start_code、extension_start_code_identifier、copyright_extension_present_flag、copyright_flag、copyright_identifier、original_or_copy、copyright_number_1、copyright_number_2、及びcopyright_number_3から構成される。

【0244】extension_start_codeは、コピーライトエクステンションのスタートを示す開始コードである。extension_start_code_identifierのどのエクステンションデータが送られるかを示すコードである。copyright_extension_present_flagは、このコピーライトエクステンション内のデータエレメントが有効か無効かを示すためのデータである。copyright_flagは、次のコピーライトエクステンション又はシーケンスエンドまで、符号化されたビデオデータに対してコピー権が与えられているか否かを示す。

【0245】copyright_identifierは、ISO/IEC JTC/SC29によって指定されたコピー権の登録機関を識別するためのデータである。original_or_copyは、ビットストリーム中のデータが、オリジナルデータであるかコピーデータであるかを示すデータである。copyright_number_1は、コピーライトナンバーのビット44から63を表わすデータである。copyright_number_2は、コピーライトナンバーのビット22から43を表わすデータである。copyright_number_3は、コピーライトナンバーのビット0から21を表わすデータである。

【0246】続いて、最終段の符号化処理によって生成されたビットストリームのピクチャ層のユーザエリアには、過去の符号化処理において使用されたピクチャディスプレイエクステンション（picture_display_extension）が、履歴ストリームとして記述されている。

【0247】このピクチャディスプレイエクステンションを表わすデータエレメントは、extension_start_code、extension_start_code_identifier、picture_display_extension_present_flag、frame_center_horizontal_offset_1、frame_center_vertical_offset_1、frame_center_horizontal_offset_2、frame_center_vertical_offset_2、frame_center_horizontal_offset_3、及びframe_center_vertical_offset_3から構成される。

【0248】extension_start_codeは、ピクチャディスプレイエクステンションのスタートを示すための開始コードである。extension_start_code_identifierは、ど

の拡張データが送られるかを示すコードである。picture_display_extension_present_flagは、ピクチャディスプレイエクステンション内のデータエレメントが有効か無効かを示すデータである。frame_center_horizontal_offsetは、表示エリアの水平方向のオフセットを示すデータであって、3つのオフセット値まで定義することができる。frame_center_vertical_offsetは、表示エリアを垂直方向のオフセットを示すデータであって、3つのオフセット値まで定義することができる。

【0249】最終段の符号化処理において生成されたビットストリームのピクチャ層のユーザエリアには、既に説明したピクチャディスプレイエクステンションを表わす履歴情報の次に、過去の符号化処理において使用されたユーザデータが、履歴ストリームとして記述されている。

【0250】このユーザデータの次には、過去の符号化処理において使用されたマクロブロック層に関する情報が、履歴ストリームとして記述されている。

【0251】このマクロブロック層に関する情報は、macroblock_address_h、macroblock_address_v、slice_header_present_flag、skipped_macroblock_flag等のマクロブロックの位置に関するデータエレメントと、macroblock_quant、macroblock_motion_forward、macroblock_motion_backward、macroblock_pattern、macroblock_intra、spatial_temporal_weight_code_flag、frame_motion_type、及びdct_type等のマクロブロックモードに関するデータエレメントと、quantiser_scale_code等の量子化ステップ制御に関するデータエレメントと、PMV[0][0]、PMV[0][0][1]、motion_vertical_field_select[0][0]、PMV[0][1][0]、PMV[0][1][1]、motion_vertical_field_select[0][1]、PMV[1][0][0]、PMV[1][0][1]、motion_vertical_field_select[1][0]、PMV[1][1][0]、PMV[1][1][1]、motion_vertical_field_select[1][1]等の動き補償に関するデータエレメントと、coded_block_pattern等のマクロブロックパターンに関するデータエレメントと、num_mv_bits、num_coef_bits、及びnum_other_bits等の発生符号量に関するデータエレメントから構成されている。

【0252】以下にマクロブロック層に関するデータエレメントについて詳細に説明する。

【0253】macroblock_address_hは、現在のマクロブロックの水平方向の絶対位置を定義するためのデータである。macroblock_address_vは、現在のマクロブロックの垂直方向の絶対位置を定義するためのデータである。slice_header_present_flagは、このマクロブロックがスライス層の先頭であり、スライスヘッダを伴うか否かを示すデータである。skipped_macroblock_flagは、復号化処理においてこのマクロブロックをスキップするか否かを示すデータである。

【0254】macroblock_quantは、後述する図65乃至

図67に示されたマクロブロックタイプ（macroblock_type）から導かれるデータであって、quantiser_scale_codeがビットストリーム中に現れるか否かを示すデータである。macroblock_motion_forwardは、図65乃至図67に示されたマクロブロックタイプから導かれるデータであって、復号化処理で使用されるデータである。macroblock_motion_backwardは、図65乃至図67に示されたマクロブロックタイプから導かれるデータであって、復号化処理で使用されるデータである。macroblock_patternは、図65乃至図67に示されたマクロブロックタイプから導かれるデータであって、coded_block_patternがビットストリーム中に現れるか否かを示すデータである。

【0255】macroblock_intraは、図65乃至図67に示されたマクロブロックタイプから導かれるデータであって、復号化処理で使用されるデータである。spatial_temporal_weight_code_flagは、図65乃至図67に示されたマクロブロックタイプから導かれるデータであって、時間スケラビリティで下位レイヤ画像のアップサンプリング方法を示すspatial_temporal_weight_codeは、ビットストリーム中に存在するか否かを示すデータである。

【0256】frame_motion_typeは、フレームのマクロブロックの予測タイプを示す2ビットのコードである。予測ベクトルが2個でフィールドベースの予測タイプであれば「00」であって、予測ベクトルが1個でフィールドベースの予測タイプであれば「01」であって、予測ベクトルが1個でフレームベースの予測タイプであれば「10」であって、予測ベクトルが1個でディアルプライムの予測タイプであれば「11」である。field_motion_typeは、フィールドのマクロブロックの動き予測を示す2ビットのコードである。予測ベクトルが1個でフィールドベースの予測タイプであれば「01」であって、予測ベクトルが2個で18×8マクロブロックベースの予測タイプであれば「10」であって、予測ベクトルが1個でディアルプライムの予測タイプであれば「11」である。dct_typeは、DCTがフレームDCTモードか、フィールドDCTモードかを示すデータである。quantiser_scale_codeはマクロブロックの量子化ステップサイズを示すデータである。

【0257】次に動きベクトルに関するデータエレメントについて説明する。動きベクトルは、復号時に必要な動きベクトルを減少させるために、先に符号化されたベクトルに関し差分として符号化される。動きベクトルの復号を行うために復号器は、4個の動きベクトル予測値（それぞれ水平及び垂直成分を伴う）を維持しなければいけない。この予測動きベクトルをPMV[r][s][v]と表わすことにしている。[r]は、マクロブロックにおける動きベクトルが第1のベクトルであるのか、第2のベクトルであるのかを示すフラグであって、マクロブロック

におけるベクトルが第1のベクトルである場合には「0」となって、マクロブロックにおけるベクトルが第2のベクトルである場合には「1」となる。[s]は、マクロブロックにおける動きベクトルの方向が、前方向であるのか後方向であるのかを示すフラグであって、前方向動きベクトルの場合には「0」となって、後方向動きベクトルの場合には「1」となる。[v]は、マクロブロックにおけるベクトルの成分が、水平方向であるのか垂直方向であるのかを示すフラグであって、水平方向成分の場合には「0」となって、垂直方向成分の場合には「1」となる。

【0258】従って、PMV[0][0][0]は、第1のベクトルの前方向の動きベクトルの水平方向成分のデータを表わし、PMV[0][0][1]は、第1のベクトルの前方向の動きベクトルの垂直方向成分のデータを表わし、PMV[0][1][0]は、第1のベクトルの後方向の動きベクトルの水平方向成分のデータを表わし、PMV[0][1][1]は、第1のベクトルの後方向の動きベクトルの垂直方向成分のデータを表わし、PMV[1][0][0]は、第2のベクトルの前方向の動きベクトルの水平方向成分のデータを表わし、PMV[1][0][1]は、第2のベクトルの前方向の動きベクトルの垂直方向成分のデータを表わし、PMV[1][1][0]は、第2のベクトルの後方向の動きベクトルの水平方向成分のデータを表わし、PMV[1][1][1]は、第2のベクトルの後方向の動きベクトルの垂直方向成分のデータを表わしている。

【0259】motion_vertical_field_select[r][s]は、予測の形式にいずれの参照フィールドを使用するかを示すデータである。このmotion_vertical_field_select[r][s]が「0」の場合には、トップ参照フィールドを使用し、「1」の場合には、ボトム参照フィールドを使用することを示している。

【0260】よって、motion_vertical_field_select[0][0]は、第1のベクトルの前方向の動きベクトルを生成する際の参照フィールドを示し、motion_vertical_field_select[0][1]は、第1のベクトルの後方向の動きベクトルを生成する際の参照フィールドを示し、motion_vertical_field_select[1][0]は、第2のベクトルの前方向の動きベクトルを生成する際の参照フィールドを示し、motion_vertical_field_select[1][1]は、第2ベクトルの後方向の動きベクトルを生成する際の参照フィールドを示している。

【0261】coded_block_patternは、DCT係数を格納する複数のDCTブロックのうち、どのDCTブロックに、有意係数（非0係数）があるかを示す可変長のデータである。num_mv_bitsは、マクロブロック中の動きベクトルの符号量を示すデータである。num_coef_bitsは、マクロブロック中のDCT係数の符号量を示すデータである。num_other_bitsは、マクロブロックの符号量で、動きベクトル及びDCT係数以外の符号量を示すデ

ータである。

【0262】次に、可変長の履歴ストリームから各データエレメントをデコードするためのシンタックスについて、図47乃至図64を参照して説明する。

【0263】この可変長の履歴ストリームは、next_start_code()関数、sequence_header()関数、sequence_extension()関数、extension_and_user_data(0)関数、group_of_picture_header()関数、extension_and_user_data(1)関数、picture_header()関数、picture_coding_extension()関数、extension_and_user_data(2)関数、及びpicture_data()関数によって定義されたデータエレメントによって構成される。

【0264】next_start_code()関数は、ビットストリーム中に存在するスタートコードを探すための関数であるので、履歴ストリームの最も先頭には、図48に示すような、過去の符号化処理において使用されたデータエレメントであってsequence_header()関数によって定義されたデータエレメントが記述されている。

【0265】sequence_header()関数によって定義されたデータエレメントは、sequence_header_code、sequence_header_present_flag、horizontal_size_value、vertical_size_value、aspect_ratio_information、frame_rate_code、bit_rate_value、marker_bit、VBV_buffer_size_value、constrained_parameter_flag、load_intra_quantizer_matrix、intra_quantizer_matrix、load_non_intra_quantizer_matrix、及びnon_intra_quantizer_matrix等である。

【0266】sequence_header_codeは、シーケンス層のスタート同期コードを表すデータである。sequence_header_present_flagは、sequence_header内のデータが有効か無効かを示すデータである。horizontal_size_valueは、画像の水平方向の画素数の下位12ビットから成るデータである。vertical_size_valueは、画像の縦のライン数の下位12ビットからなるデータである。aspect_ratio_informationは、画素のアスペクト比（縦横比）または表示画面アスペクト比を表すデータである。frame_rate_codeは、画像の表示周期を表すデータである。bit_rate_valueは、発生ビット量に対する制限のためのビット・レートの下位18ビット（400bsp単位で切り上げる）データである。

【0267】marker_bitは、スタートコードエミュレーションを防止するために挿入されるビットデータである。VBV_buffer_size_valueは、発生符号量制御用の仮想バッファ（ビデオバッファペリファイヤー）の大きさを決める値の下位10ビットデータである。constrained_parameter_flagは、各パラメータが制限以内であることを示すデータである。load_intra_quantizer_matrixは、イントラMB用量子化マトリックス・データの存在を示すデータである。intra_quantizer_matrixは、イントラMB用量子化マトリックスの値を示すデータである。

load_non_intra_quantizer_matrixは、非イントラMB用量子化マトリックス・データの存在を示すデータである。non_intra_quantizer_matrixは、非イントラMB用量子化マトリックスの値を表すデータである。

【0268】sequence_header()関数によって定義されたデータ要素の次には、図49で示すような、sequence_extension()関数によって定義されたデータ要素が、履歴ストリームとして記述されている。

【0269】sequence_extension()関数によって定義されたデータ要素とは、extension_start_code、extension_start_code_identifier、sequence_extension_present_flag、profile_and_level_indication、progressive_sequence、chroma_format、horizontal_size_extension、vertical_size_extension、bit_rate_extension、vbv_buffer_size_extension、low_delay、frame_rate_extension_n、及びframe_rate_extension_d等のデータ要素である。

【0270】extension_start_codeは、エクステンションデータのスタート同期コードを表すデータである。extension_start_code_identifierは、どの拡張データが送られるかを示すデータである。sequence_extension_present_flagは、シーケンスエクステンション内のデータが有効であるか無効であるかを示すデータである。profile_and_level_indicationは、ビデオデータのプロファイルとレベルを指定するためのデータである。progressive_sequenceは、ビデオデータが順次走査であることを示すデータである。chroma_formatは、ビデオデータの色差フォーマットを指定するためのデータである。horizontal_size_extensionは、シーケンスヘッダのhorizontal_size_valueに加える上位2ビットのデータである。vertical_size_extensionは、シーケンスヘッダのvertical_size_valueに加える上位2ビットのデータである。bit_rate_extensionは、シーケンスヘッダのbit_rate_valueに加える上位12ビットのデータである。vbv_buffer_size_extensionは、シーケンスヘッダのvbv_buffer_size_valueに加える上位8ビットのデータである。

【0271】low_delayは、Bピクチャを含まないことを示すデータである。frame_rate_extension_nは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを求めるためのデータである。frame_rate_extension_dは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを求めるためのデータである。

【0272】sequence_extension()関数によって定義されたデータ要素の次には、図50に示すようなextension_and_user_data(0)関数によって定義されたデータ要素が、履歴ストリームとして記述されている。extension_and_user_data(i)関数は、「i」が2以外のときは、extension_data()関数によって定義されるデータ要素は記述せずに、user_data()関数によって定義されるデータ要素のみを履歴ストリーム

として記述する。よって、extension_and_user_data(0)関数は、user_data()関数によって定義されるデータ要素のみを履歴ストリームとして記述する。

【0273】user_data()関数は、図51に示されたようなシンタックスに基いて、ユーザデータを履歴ストリームとして記述する。

【0274】extension_and_user_data(0)関数によって定義されたデータ要素の次には、図52に示すようなgroup_of_picture_header()関数によって定義されたデータ要素、及びextension_and_user_data(1)関数によって定義されるデータ要素が、履歴ストリームとして記述されている。但し、履歴ストリーム中に、GOP層のスタートコードを示すgroup_start_codeが記述されている場合にのみ、group_of_picture_header()関数によって定義されたデータ要素、及びextension_and_user_data(1)関数によって定義されるデータ要素が記述されている。

【0275】group_of_picture_header()関数によって定義されたデータ要素は、group_start_code、group_of_picture_header_present_flag、time_code、closed_gop、及びbroken_linkから構成される。

【0276】group_start_codeは、GOP層の開始同期コードを示すデータである。group_of_picture_header_present_flagは、group_of_picture_header内のデータ要素が有効であるか無効であるかを示すデータである。time_codeは、GOPの先頭ピクチャのシーケンスの先頭からの時間を示すタイムコードである。closed_gopは、GOP内の画像が他のGOPから独立再生可能なことを示すフラグデータである。broken_linkは、編集などのためにGOP内の先頭のBピクチャが正確に再生できないことを示すフラグデータである。

【0277】extension_and_user_data(1)関数は、extension_and_user_data(0)関数と同じように、user_data()関数によって定義されるデータ要素のみを履歴ストリームとして記述する。

【0278】もし、履歴ストリーム中に、GOP層のスタートコードを示すgroup_start_codeが存在しない場合には、これらのgroup_of_picture_header()関数及びextension_and_user_data(1)関数によって定義されるデータ要素は、履歴ストリーム中には記述されていない。その場合には、extension_and_user_data(0)関数によって定義されたデータ要素の次に、picture_header()関数によって定義されたデータ要素が履歴ストリームとして記述されている。

【0279】picture_header()関数によって定義されたデータ要素は、図53に示すように、picture_start_code、temporal_reference、picture_coding_type、vbv_delay、full_pel_forward_vector、forward_f_code、full_pel_backward_vector、backward_f_code、extra_bit_picture、及びextra_information_pictureであ

る。

【0280】具体的には、picture_start_codeは、ピクチャ層の開始同期コードを表すデータである。temporal_referenceは、ピクチャの表示順を示す番号でGOPの先頭でリセットされるデータである。picture_coding_typeは、ピクチャタイプを示すデータである。vbv_delayは、ランダムアクセス時の仮想バッファの初期状態を示すデータである。full_pel_forward_vectorは、順方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。forward_f_codeは、順方向動きベクトル探索範囲を表すデータである。full_pel_backward_vectorは、逆方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。backward_f_codeは、逆方向動きベクトル探索範囲を表すデータである。extra_bit_pictureは、後続する追加情報の存在を示すフラグである。このextra_bit_pictureが「1」の場合には、次にextra_information_pictureが存在し、extra_bit_pictureが「0」の場合には、これに続くデータが無いことを示している。extra_information_pictureは、規格において予約された情報である。

【0281】picture_headr()関数によって定義されたデータエレメントの次には、図54に示すようなpicture_coding_extension()関数によって定義されたデータエレメントが、履歴ストリームとして記述されている。

【0282】このpicture_coding_extension()関数によって定義されたデータエレメントとは、extension_start_code、extension_start_code_identifier、f_code[0][0]、f_code[0][1]、f_code[1][0]、f_code[1][1]、intra_dc_precision、picture_structure、top_field_first、frame_predictive_frame_dct、concealment_motion_vectors、q_scale_type、intra_vlc_format、alternate_scan、repeat_firt_field、chroma_420_type、progressive_frame、composite_display_flag、v_axis、field_sequence、sub_carrier、burst_amplitude、及びsub_carrier_phaseから構成される。

【0283】extension_start_codeは、ピクチャ層のエクステンションデータのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。f_code[0][0]は、フォワード方向の水平動きベクトル探索範囲を表すデータである。f_code[0][1]は、フォワード方向の垂直動きベクトル探索範囲を表すデータである。f_code[1][0]は、バックワード方向の水平動きベクトル探索範囲を表すデータである。f_code[1][1]は、バックワード方向の垂直動きベクトル探索範囲を表すデータである。intra_dc_precisionは、DC係数の精度を表すデータである。

【0284】picture_structureは、フレームストラクチャかフィールドストラクチャかを示すデータである。フィールドストラクチャの場合は、上位フィールドか下

位フィールドかもあわせて示すデータである。top_field_firstは、フレームストラクチャの場合、最初のフィールドが上位か下位かを示すデータである。frame_predictive_frame_dctは、フレーム・ストラクチャの場合、フレーム・モードDCTの予測がフレーム・モードだけであることを示すデータである。concealment_motion_vectorsは、イントラマクロブロックに伝送エラーを隠蔽するための動きベクトルがついていることを示すデータである。q_scale_typeは、線形量子化スケールを利用するか、非線形量子化スケールを利用するかを示すデータである。intra_vlc_formatは、イントラマクロブロックに、別の2次元VLCを使うかどうかを示すデータである。

【0285】alternate_scanは、ジグザグスキャンを使うか、オルタネート・スキャンを使うかの選択を表すデータである。repeat_firt_fieldは、2:3プルダウンの際に使われるデータである。chroma_420_typeは、信号フォーマットが4:2:0の場合、次のprogressive_frameと同じ値、そうでない場合は0を表すデータである。progressive_frameは、このピクチャが、順次走査できているかどうかを示すデータである。composite_display_flagは、ソース信号がコンポジット信号であったかどうかを示すデータである。v_axisは、ソース信号が、PALの場合に使われるデータである。field_sequenceは、ソース信号が、PALの場合に使われるデータである。sub_carrierは、ソース信号が、PALの場合に使われるデータである。burst_amplitudeは、ソース信号が、PALの場合に使われるデータである。sub_carrier_phaseは、ソース信号が、PALの場合に使われるデータである。

【0286】picture_coding_extension()関数によって定義されたデータエレメントの次には、extensions_and_user_data(2)によって定義されたデータエレメントが、履歴ストリームとして記述されている。このextension_and_user_data(2)関数は、図50に示したように、ビットストリーム中にエクステンションスタートコード(extension_start_code)が存在する場合には、extension_data()関数によって定義されるデータエレメントが記述されている。このデータエレメントの次には、ビットストリーム中にユーザデータスタートコード(user_data_start_code)が存在する場合には、user_data()関数によって定義されるデータエレメントが記述されている。但し、ビットストリーム中にエクステンションスタートコード及びユーザデータスタートコードが存在しない場合にはextension_data()関数及びuser_data()関数によって定義されるデータエレメントはビットストリーム中には記述されていない。

【0287】extension_data()関数は、図55に示すように、extension_start_codeを示すデータエレメントと、quant_matrix_extension()関数、copyright_extens

ion()関数、及びpicture_display_extension()関数によって定義されるデータエレメントとを、ビットストリーム中に履歴ストリームとして記述するための関数である。

【0288】quant_matrix_extension()関数によって定義されるデータエレメントは、図56に示すように、extension_start_code、extension_start_code_identifier、quant_matrix_extension_present_flag、load_intra_quantizer_matrix、intra_quantizer_matrix[64]、load_non_intra_quantizer_matrix、non_intra_quantizer_matrix[64]、load_chroma_intra_quantizer_matrix、chroma_intra_quantizer_matrix[64]、load_chroma_non_intra_quantizer_matrix、及びchroma_non_intra_quantizer_matrix[64]である。

【0289】extension_start_codeは、この量子化マトリックスエクステンションのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。quant_matrix_extension_present_flagは、この量子化マトリックスエクステンション内のデータエレメントが有効か無効かを示すためのデータである。load_intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。intra_quantizer_matrixは、イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【0290】load_non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスデータの存在を示すデータである。non_intra_quantizer_matrixは、非イントラマクロブロック用の量子化マトリックスの値を表すデータである。load_chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックス・データの存在を示すデータである。chroma_intra_quantizer_matrixは、色差イントラマクロブロック用の量子化マトリックスの値を示すデータである。load_chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックス・データの存在を示すデータである。chroma_non_intra_quantizer_matrixは、色差非イントラマクロブロック用の量子化マトリックスの値を示すデータである。

【0291】copyright_extension()関数によって定義されるデータエレメントは、図57に示すように、extension_start_code、extension_start_code_identifier、copyright_extension_present_flag、copyright_flag、copyright_identifier、original_or_copy、copyright_number_1、copyright_number_2、及びcopyright_number_3から構成される。

【0292】extension_start_codeは、コピーライトエクステンションのスタートを示す開始コードである。extension_start_code_identifierどのエクステンションデータが送られるかを示すコードである。copyright_ext

ension_present_flagは、このコピーライトエクステンション内のデータエレメントが有効か無効かを示すためのデータである。

【0293】copyright_flagは、次のコピーライトエクステンション又はシーケンスエンドまで、符号化されたビデオデータに対してコピー権が与えられているか否かを示す。copyright_identifierは、ISO/IEC JTC/SC29によって指定されたコピー権の登録機関を識別するためのデータである。original_or_copyは、ビットストリーム中のデータが、オリジナルデータであるかコピーデータであるかを示すデータである。copyright_number_1は、コピーライトナンバーのビット44から63を表わすデータである。copyright_number_2は、コピーライトナンバーのビット22から43を表わすデータである。copyright_number_3は、コピーライトナンバーのビット0から21を表わすデータである。

【0294】picture_display_extension()関数によって定義されるデータエレメントは、図58に示すように、extension_start_code_identifier、frame_center_horizontal_offset、frame_center_vertical_offset等である。

【0295】extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。frame_center_horizontal_offsetは、表示エリアの水平方向のオフセットを示すデータであって、number_of_frame_center_offsetsによって定義される数のオフセット値を定義することができる。frame_center_vertical_offsetは、表示エリアを垂直方向のオフセットを示すデータであって、number_of_frame_center_offsetsによって定義される数のオフセット値を定義することができる。

【0296】再び図47に戻って、extension_and_user_data(2)関数によって定義されるデータエレメントの次には、picture_data()関数によって定義されるデータエレメントが、履歴ストリームとして記述されている。

【0297】picture_data()関数によって定義されるデータエレメントは、図59に示すように、slice()関数によって定義されるデータエレメントである。但し、ビットストリーム中に、slice()関数のスタートコードを示すslice_start_codeが存在しない場合には、このslice()関数によって定義されるデータエレメントはビットストリーム中に記述されていない。

【0298】slice()関数は、図60に示されるように、slice_start_code、slice_quantiser_scale_code、intra_slice_flag、intra_slice、reserved_bits、extra_bit_slice、extra_information_slice、及びextra_bit_slice等のデータエレメントと、macroblock()関数によって定義されるデータエレメントを、履歴ストリームとして記述するための関数である。

【0299】slice_start_codeは、slice()関数によって定義されるデータエレメントのスタートを示すスター

トコードである。slice_quantiser_scale_codeは、このスライス層に存在するマクロブロックに対して設定された量子化ステップサイズを示すデータである。しかし、各マクロブロック毎に、quantiser_scale_codeが設定されている場合には、各マクロブロックに対して設定されたmacroblock_quantiser_scale_codeのデータが優先して使用される。

【0300】intra_slice_flagは、ビットストリーム中にintra_slice及びreserved_bitsが存在するかを示すフラグである。intra_sliceは、スライス層中にノンイントラマクロブロックが存在するかを示すデータである。スライス層におけるマクロブロックのいずれかがノンイントラマクロブロックである場合には、intra_sliceは「0」となり、スライス層におけるマクロブロックの全てがノンイントラマクロブロックである場合には、intra_sliceは「1」となる。reserved_bitsは、7ビットのデータであって「0」の値を取る。extra_bit_sliceは、履歴ストリームとして追加の情報が存在することを示すフラグであって、次にextra_information_sliceが存在する場合には「1」に設定される。追加の情報が存在しない場合には「0」に設定される。

【0301】これらのデータエレメントの次には、macroblock()関数によって定義されたデータエレメントが、履歴ストリームとして記述されている。

【0302】macroblock()関数は、図61に示すように、macroblock_escape、macroblock_address_increment、及びmacroblock_quantiser_scale_code等のデータエレメントと、macroblock_modes()関数、及びmacroblock_vectors(s)関数によって定義されたデータエレメントを記述するための関数である。

【0303】macroblock_escapeは、参照マクロブロックと前のマクロブロックとの水平方向の差が34以上であるか否かを示す固定ビット列である。参照マクロブロックと前のマクロブロックとの水平方向の差が34以上の場合には、macroblock_address_incrementの値に33をプラスする。macroblock_address_incrementは、参照マクロブロックと前のマクロブロックとの水平方向の差を示すデータである。もし、このmacroblock_address_incrementの前にmacroblock_escapeが1つ存在するのであれば、このmacroblock_address_incrementの値に33をプラスした値が、実際の参照マクロブロックと前のマクロブロックとの水平方向の差分を示すデータとなる。

【0304】macroblock_quantiser_scale_codeは、各マクロブロック毎に設定された量子化ステップサイズである。各スライス層には、スライス層の量子化ステップサイズを示すslice_quantiser_scale_codeが設定されているが、参照マクロブロックに対してmacroblock_quantiser_scale_codeが設定されている場合には、この量子化ステップサイズを選択する。

【0305】macroblock_address_incrementの次には、

macroblock_modes()関数によって定義されるデータエレメントが記述されている。macroblock_modes()関数は、図62に示すように、macroblock_type、frame_motion_type、field_motion_type、dct_type等のデータエレメントを、履歴ストリームとして記述するための関数である。

【0306】macroblock_typeは、マクロブロックの符号化タイプを示すデータである。具体的には、図65乃至図67に示されるように、macroblock_typeは、macroblock_quant、dct_type_flag、macroblock_motion_forward、及びmacroblock_motion_backwardなどのフラグから生成された可変長データである。macroblock_quantは、マクロブロックに対して量子化ステップサイズを設定するためのmacroblock_quantiser_scale_codeが設定されているか否かを示すフラグであって、ビットストリーム中にmacroblock_quantiser_scale_codeが存在する場合には、macroblock_quantは「1」の値を取る。

【0307】dct_type_flagは、参照マクロブロックがフレームDCT又はフィールドDCTで符号化されているかを示すdct_typeが存在するか否かを示すためのフラグ（言い換えるとDCTされているか否かを示すフラグ）であって、ビットストリーム中にdct_typeが存在する場合には、このdct_type_flagは「1」の値を取る。macroblock_motion_forwardは、参照マクロブロックが前方予測されているか否かを示すフラグであって、前方予測されている場合には「1」の値を取る。macroblock_motion_backwardは、参照マクロブロックが後方予測されているか否かを示すフラグであって、後方予測されている場合には「1」の値を取る。

【0308】もし、macroblock_motion_forward又はmacroblock_motion_backwardが「1」のときに、ピクチャ構造がフレームのときに、frame_period_frame_dctが「0」のときには、macroblock_typeを表わすデータエレメントの次にframe_motion_typeを表わすデータエレメントが記述されている。尚、このframe_period_frame_dctは、frame_motion_typeがビットストリーム中に存在するか否かを示すフラグである。

【0309】frame_motion_typeは、フレームのマクロブロックの予測タイプを示す2ビットのコードである。予測ベクトルが2個でフィールドベースの予測タイプであれば「00」であって、予測ベクトルが1個でフィールドベースの予測タイプであれば「01」であって、予測ベクトルが1個でフレームベースの予測タイプであれば「10」であって、予測ベクトルが1個でディアルプライムの予測タイプであれば「11」である。

【0310】もし、macroblock_motion_forward又はmacroblock_motion_backwardが「1」のときに、ピクチャ構造がフレームでない場合には、macroblock_typeを表わすデータエレメントの次にfield_motion_typeを表わすデータエレメントが記述されている。

【0311】field_motion_typeは、フィールドのマクロブロックの動き予測を示す2ビットのコードである。予測ベクトルが1個でフィールドベースの予測タイプであれば「01」であって、予測ベクトルが2個で18×8マクロブロックベースの予測タイプであれば「10」であって、予測ベクトルが1個でディアルプライムの予測タイプであれば「11」である。

【0312】もし、ピクチャ構造がフレームで、frame_period_frame_dctがframe_motion_typeがビットストリーム中に存在することを示し、且つ、frame_period_frame_dctがdct_typeがビットストリーム中に存在することを示す場合には、macroblock_typeを表わすデータエレメントの次にdct_typeを表わすデータエレメントが記述されている。尚、dct_typeは、DCTがフレームDCTモードか、フィールドDCTモードかを示すデータである。

【0313】再び図61に戻って、もし、参照マクロブロックが前方予測マクロブロックであるか又は参照マクロブロックがイントラマクロブロックであって且つコンシール処理のマクロブロックのいずれかの場合には、motion_vectors(0)関数によって定義されるデータエレメントが記述される。また、参照マクロブロックが後方予測マクロブロックである場合には、motion_vectors(1)関数によって定義されるデータエレメントが記述される。尚、motion_vectors(0)関数は、第1番めの動きベクトルに関するデータエレメントを記述するための関数であって、motion_vectors(1)関数は、第2番めの動きベクトルに関するデータエレメントを記述するための関数である。

【0314】motion_vectors(s)関数は、図63に示されるように、動きベクトルに関するデータエレメントを記述するための関数である。

【0315】もし、動きベクトルが1個でディアルプライム予測モードを使用していない場合には、motion_vertical_field_select[0][s]とmotion_vector(0,s)によって定義されるデータエレメントが記述される。

【0316】このmotion_vertical_field_select[r][s]は、第1番目の動きベクトル（前方又は後方のどちらのベクトルであっても良い）が、ボトムフィールドを参照して作られたベクトルであるかトップフィールドを参照して作られたベクトルであるかを示すフラグである。この指標“r”は、第1番めのベクトル又は第2番めのベクトルのいずれのベクトルであるかを示す指標であって、“s”は、予測方向が前方又は後方予測のいずれであるかを示す指標である。

【0317】motion_vector(r,s)関数は、図64に示されるように、motion_code[r][s][t]に関するデータ列と、motion_residual[r][s][t]に関するデータ列と、dmvector[t]を表わすデータとを記述するための関数である。

【0318】motion_code[r][s][t]は、動きベクトルの大きさを-16～+16の範囲で表わす可変長のデータである。motion_residual[r][s][t]は、動きベクトルの残差を表わす可変長のデータである。よって、このmotion_code[r][s][t]とmotion_residual[r][s][t]との値によって詳細な動きベクトルを記述することができる。dmvector[t]は、ディアルプライム予測モードのときに、一方のフィールド（例えばボトムフィールドに対してトップフィールドを一方のフィールドとする）における動きベクトルを生成するために、時間距離に応じた既存の動きベクトルがスケールされると共に、トップフィールドとボトムフィールドとのライン間の垂直方向のずれを反映させるために垂直方向に対して補正を行うデータである。この指標“r”は、第1番めのベクトル又は第2番めのベクトルのいずれのベクトルであるかを示す指標であって、“s”は、予測方向が前方又は後方予測のいずれであるかを示す指標である。“s”は、動きベクトルが垂直方向の成分であるか水平方向の成分であるかを示すデータである。

【0319】図64に示されmotion_vector(r,s)関数によって、まず、水平方向のmotion_coder[r][s][0]を表わすデータ列が、履歴ストリームとして記述される。motion_residual[0][s][t]及びmotion_residual[1][s][t]の双方のビット数は、f_code[s][t]で示されるので、f_code[s][t]が1でない場合には、motion_residual[r][s][t]がビットストリーム中に存在することを示すことになる。水平方向成分のmotion_residual[r][s][0]が「1」でなくて、水平方向成分のmotion_code[r][s][0]が「0」でないということは、ビットストリーム中にmotion_residual[r][s][0]を表わすデータエレメントが存在し、動きベクトルの水平方向成分が存在するということの意味しているので、その場合には、水平方向成分のmotion_residual[r][s][0]を表わすデータエレメントが記述されている。

【0320】続いて、垂直方向のmotion_coder[r][s][1]を表わすデータ列が、履歴ストリームとして記述される。同じようにmotion_residual[0][s][t]及びmotion_residual[1][s][t]の双方のビット数は、f_code[s][t]で示されるので、f_code[s][t]が1でない場合には、motion_residual[r][s][t]がビットストリーム中に存在することを表わすことになる。motion_residual[r][s][1]が「1」でなくて、motion_code[r][s][1]が「0」でないということは、ビットストリーム中にmotion_residual[r][s][1]を表わすデータエレメントが存在し、動きベクトルの垂直方向成分が存在するということの意味しているので、その場合には、垂直方向成分のmotion_residual[r][s][1]を表わすデータエレメントが記述されている。

【0321】なお、可変長フォーマットにおいては、伝送するビットレートを減少させるために、履歴情報を削

減することができる。

【0322】すなわち、macroblock_typeとmotion_vectors()は転送するが、quantiser_scale_codeを転送しない場合には、slice_quantiser_scale_codeを"0000"とすることで、ビットレートを減少させることができる。

【0323】また、macroblock_typeのみ転送し、motion_vectors()、quantiser_scale_code、およびdct_typeを転送しない場合には、macroblock_typeとして、"not coded"を使用することで、ビットレートを減少することができる。

【0324】さらにまた、picture_coding_typeのみ転送し、slice()以下の情報は全て転送しない場合には、slice_start_codeを持たないpicture_data()を使用することで、ビットレートを減少させることができる。

【0325】以上においては、user_data内の23ビットの連続する"0"が出ないようにする場合に、22ビット毎に"1"を挿入するようにしたが、22ビット毎でなくてもよい。また、連続する"0"の個数を数えて"1"を挿入するのではなく、Byte_alignを調べて挿入するようにすることも可能である。

【0326】さらに、MPEGにおいては、23ビットの連続する"0"の発生を禁止しているが、実際には、バイトの先頭から23ビット連続する場合だけが問題とされ、バイトの先頭ではなく、途中から0が23ビット連続する場合は、問題とされない。従って、例えば24ビット毎に、LSB以外の位置に"1"を挿入するようにしてもよい。

【0327】また、以上においては、履歴情報を、video elementary streamに近い形式にしたが、packetized elementary streamやtransport streamに近い形式にしてもよい。また、Elementary Streamのuser_dataの場所を、picture_dataの前としたが、他の場所にすることもできる。

【0328】なお、上記各処理を行うコンピュータプログラムは、磁気ディスク、CD-ROM等の情報記録媒体よりなる提供媒体のほか、インターネット、デジタル衛星などのネットワーク提供媒体を介してユーザに提供することができる。

【0329】

【発明の効果】以上の如く請求項1に記載の復号装置、請求項2に記載の復号方法、および請求項3に記載の提供媒体によれば、ビットストリームのピクチャ層のユーザデータエリアに挿入された符号化履歴情報を復号するようにしたので、小さい規模の装置で、再符号化に伴う画像の劣化を抑制することが可能となる。

【図面の簡単な説明】

【図1】高効率符号化の原理を説明する図である。

【図2】画像データを圧縮する場合におけるピクチャタイプを説明する図である。

【図3】画像データを圧縮する場合におけるピクチャタイプを説明する図である。

【図4】動画画像信号を符号化する原理を説明した図である。

【図5】動画画像信号を符号化し、復号する装置の構成を示すブロック図である。

【図6】フォーマット変換を説明する図である。

【図7】図5のエンコーダ18の構成を示すブロック図である。

【図8】図7の予測モード切替回路52の動作を説明する図である。

【図9】図7の予測モード切替回路52の動作を説明する図である。

【図10】図7の予測モード切替回路52の動作を説明する図である。

【図11】図7の予測モード切替回路52の動作を説明する図である。

【図12】図5のデコーダ31の構成を示すブロック図である。

【図13】ピクチャタイプに対応したSNR制御を説明する図である。

【図14】本発明を適用したトランスコーダ101の構成を示すブロック図である。

【図15】図14のトランスコーダ101のより詳細な構成を示すブロック図である。

【図16】図14の復号装置102に内蔵されるデコーダ111の構成を示すブロック図である。

【図17】マクロブロックの画素を説明する図である。

【図18】符号化パラメータが記録される領域を説明する図である。

【図19】図14の符号化装置106に内蔵されるエンコーダ121の構成を示すブロック図である。

【図20】図15のヒストリーフォーマッタ211の構成例を示すブロック図である。

【図21】図15のヒストリーデコーダ203の構成例を示すブロック図である。

【図22】図15のコンバータ212の構成例を示すブロック図である。

【図23】図22のスタッフ回路323の構成例を示すブロック図である。

【図24】図22のコンバータ212の動作を説明するタイミングチャートである。

【図25】図15のコンバータ202の構成例を示すブロック図である。

【図26】図25のディリット回路343の構成例を示すブロック図である。

【図27】図15のコンバータ212の他の構成例を示すブロック図である。

【図28】図15のコンバータ202の他の構成例を示すブロック図である。

【図29】図15のユーザデータフォーマッタ213の構成例を示すブロック図である。

【図30】図14のトランスコーダ101が実際に使用される状態を示す図である。

【図31】符号化パラメータが記録される領域を説明する図である。

【図32】図14の符号化装置106の変更可能ピクチャタイプ判定処理を説明するフローチャートである。

【図33】ピクチャタイプが変更される例を示す図である。

【図34】ピクチャタイプが変更される他の例を示す図である。

【図35】図14の符号化装置106の量子化制御処理を説明する図である。

【図36】図14の符号化装置106の量子化制御処理を説明するフローチャートである。

【図37】密結合されたトランスコーダ101の構成を示すブロック図である。

【図38】MPEGストリームのシンタックスを説明する図である。

【図39】図38のシンタックスの構成を説明する図である。

【図40】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図41】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図42】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図43】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図44】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図45】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図46】固定長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図47】可変長の履歴情報を記録するhistory_stream()のシンタックスを説明する図である。

【図48】sequence_header()のシンタックスを説明する図である。

【図49】sequence_extension()のシンタックスを説明する図である。

【図50】extension_and_user_data()のシンタックスを説明する図である。

【図51】user_data()のシンタックスを説明する図である。

【図52】group_of_pictures_header()のシンタックスを説明する図である。

【図53】picture_header()のシンタックスを説明する図である。

【図54】picture_coding_extension()のシンタックスを説明する図である。

【図55】extension_data()のシンタックスを説明する図である。

【図56】quant_matrix_extension()のシンタックスを説明する図である。

【図57】copyright_extension()のシンタックスを説明する図である。

【図58】picture_display_extension()のシンタックスを説明する図である。

【図59】picture_data()のシンタックスを説明する図である。

【図60】slice()のシンタックスを説明する図である。

【図61】macroblock()のシンタックスを説明する図である。

【図62】macroblock_modes()のシンタックスを説明する図である。

【図63】motion_vectors(s)のシンタックスを説明する図である。

【図64】motion_vector(r,s)のシンタックスを説明する図である。

【図65】Iピクチャに対するmacroblock_typeの可変長符号を説明する図である。

【図66】Pピクチャに対するmacroblock_typeの可変長符号を説明する図である。

【図67】Bピクチャに対するmacroblock_typeの可変長符号を説明する図である。

【図68】従来のトランスコーダ131の構成の一例を示すブロック図である。

【図69】従来のトランスコーダ131の構成の一例を示すブロック図である。

【図70】従来の符号化装置と復号装置の配置を説明する図である。

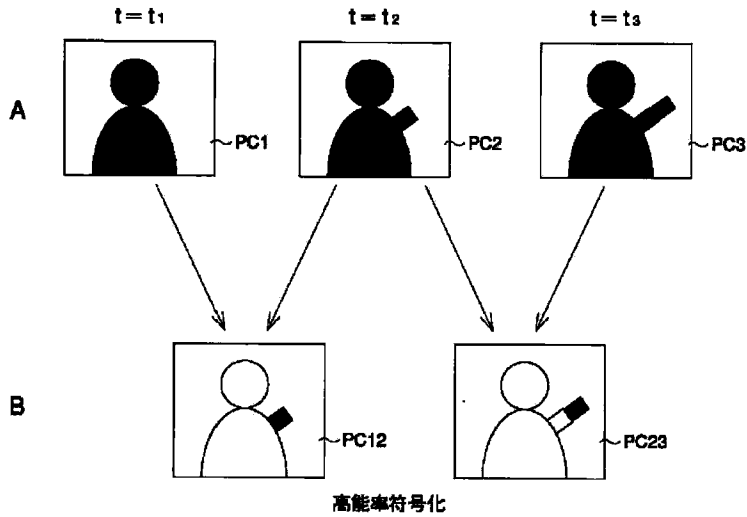
【符号の説明】

1 符号化装置, 2 復号化装置, 3 記録媒体,
12, 13 A/D変換器, 14 フレームメモリ,
15 輝度信号フレームメモリ, 16 色差信号フレームメモリ,
17 フォーマット変換回路, 18 エンコーダ, 31 デコーダ, 32 フォーマット変換回路,
33 フレームメモリ, 34 輝度信号フレームメモリ, 35 色差信号フレームメモリ, 36, 37 D/A変換器, 50 動きベクトル検出回路,
51 フレームメモリ, 52 予測モード切り替え回路, 53 演算部, 54 予測判定回路, 55 DCTモード切り替え回路, 56 DCT回路, 57 量子化回路, 58 可変長符号化回路, 59 送信バッファ,
60 逆量子化回路, 61 IDCT回路, 62 演算器, 63 フレームメモリ, 64 動き補償回路, 81 受信バッファ, 82 可変長復号化回

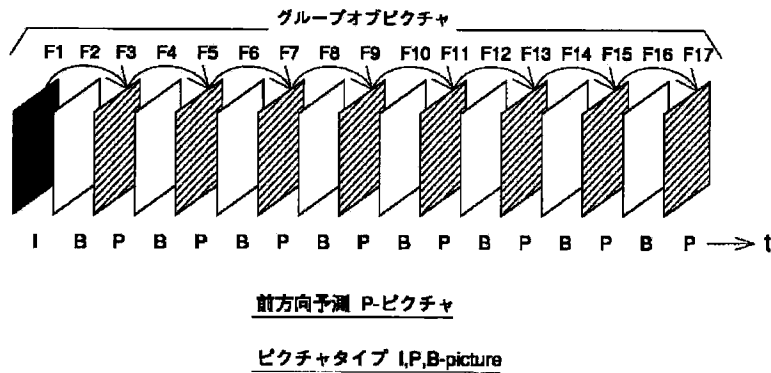
路, 83 逆量子化回路, 84 IDCT回路, 85 演算器, 86 フレームメモリ, 87 動き補償回路, 101 トランスコーダ, 102 復号装置, 103 符号化パラメータ多重装置, 105 符号化パラメータ分離装置, 106 符号化装置, 10

6 SDTI, 111 デコーダ, 112 可変長復号化回路, 121 エンコーダ, 122 符号化パラメータコントローラ, 131 トランスコーダ, 132 復号装置, 133 符号化装置, 134 動き検出部, 135 符号化部

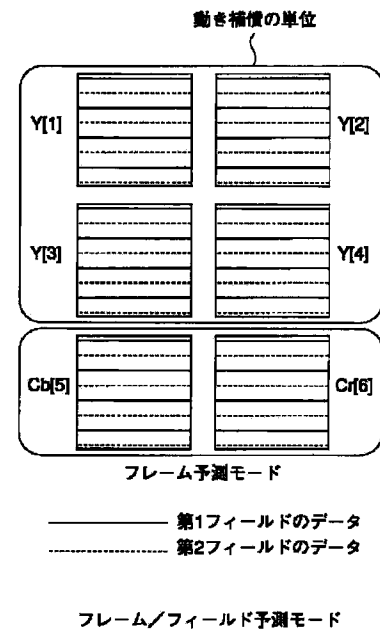
【図1】



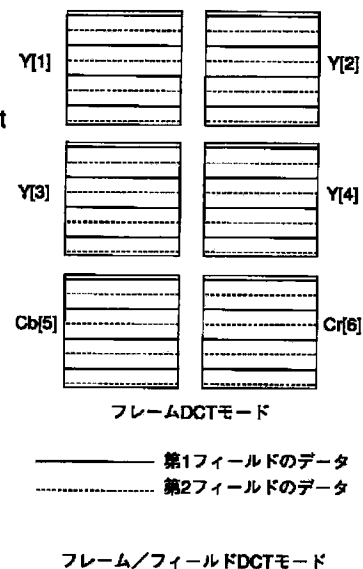
【図2】



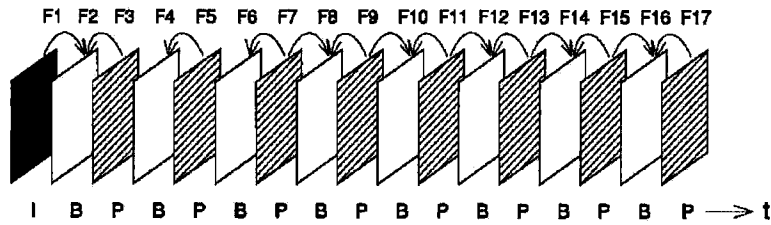
【図8】



【図10】



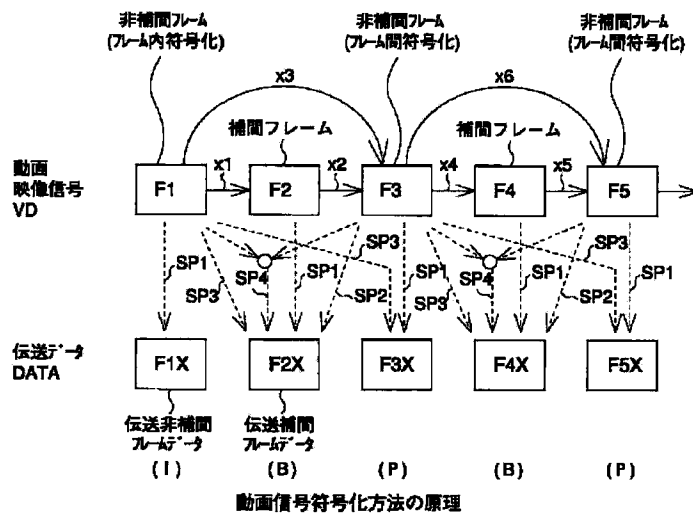
【図3】



両方向予測 B-ピクチャ

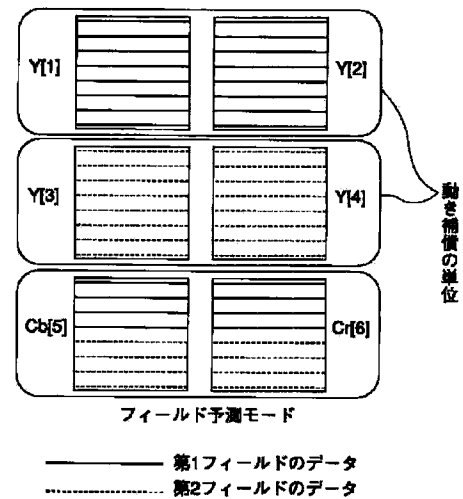
ピクチャタイプ I,P,B-picture

【図4】



動画信号符号化方法の原理

【図9】

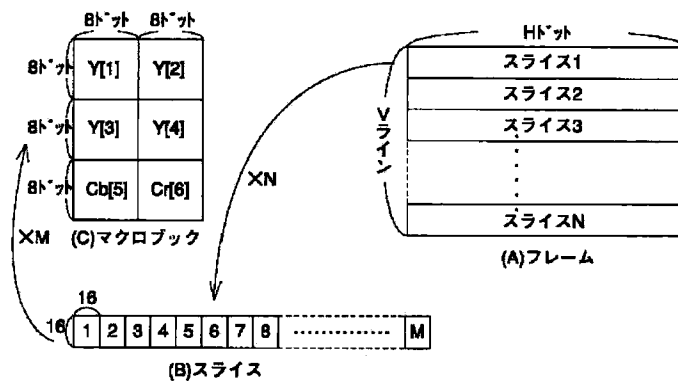


フィールド予測モード

—— 第1フィールドのデータ
 第2フィールドのデータ

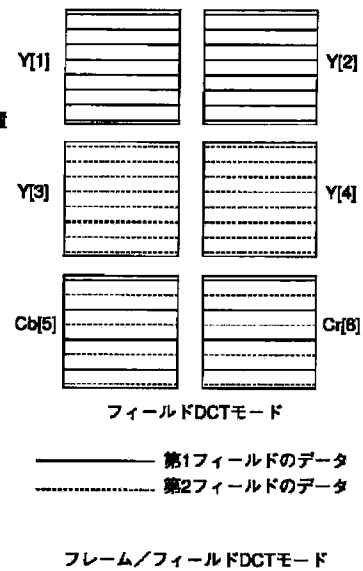
フレーム/フィールド予測モード

【図6】

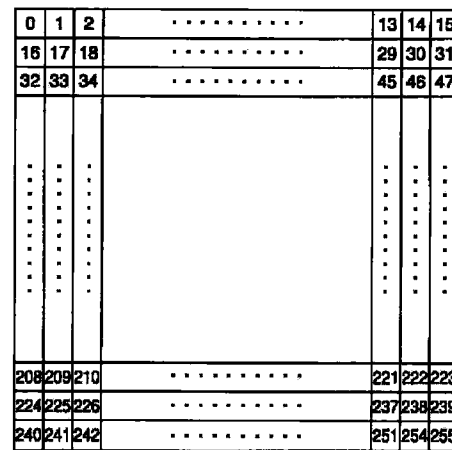


画像データの構造

【图 1-1】

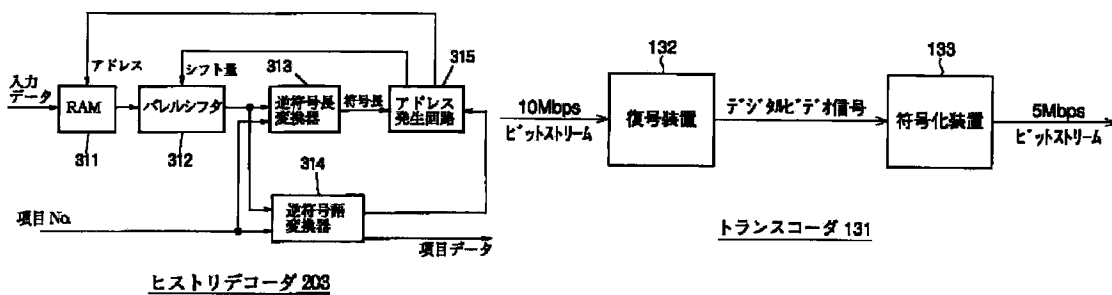


【图 17】

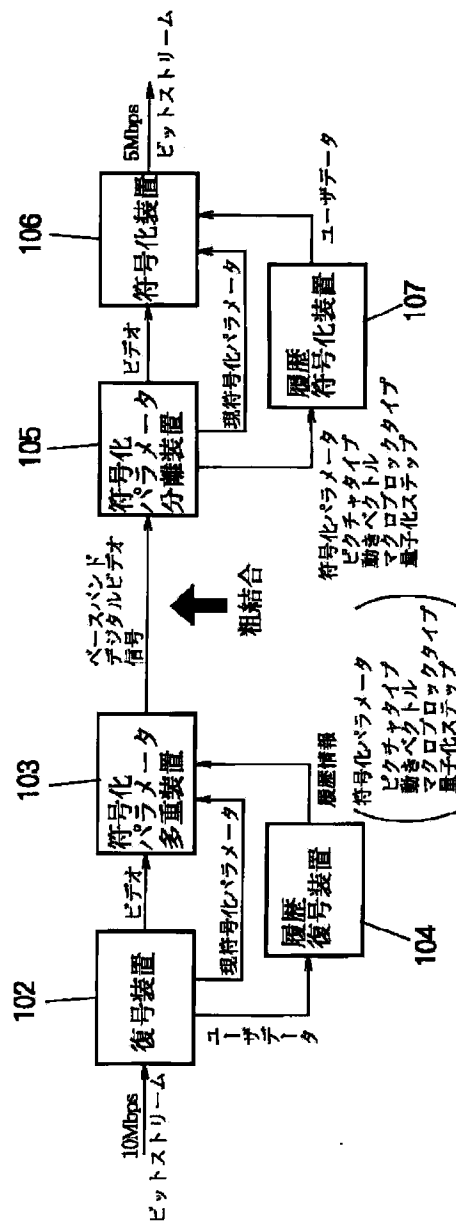


マクロブロック

【図 6 8】

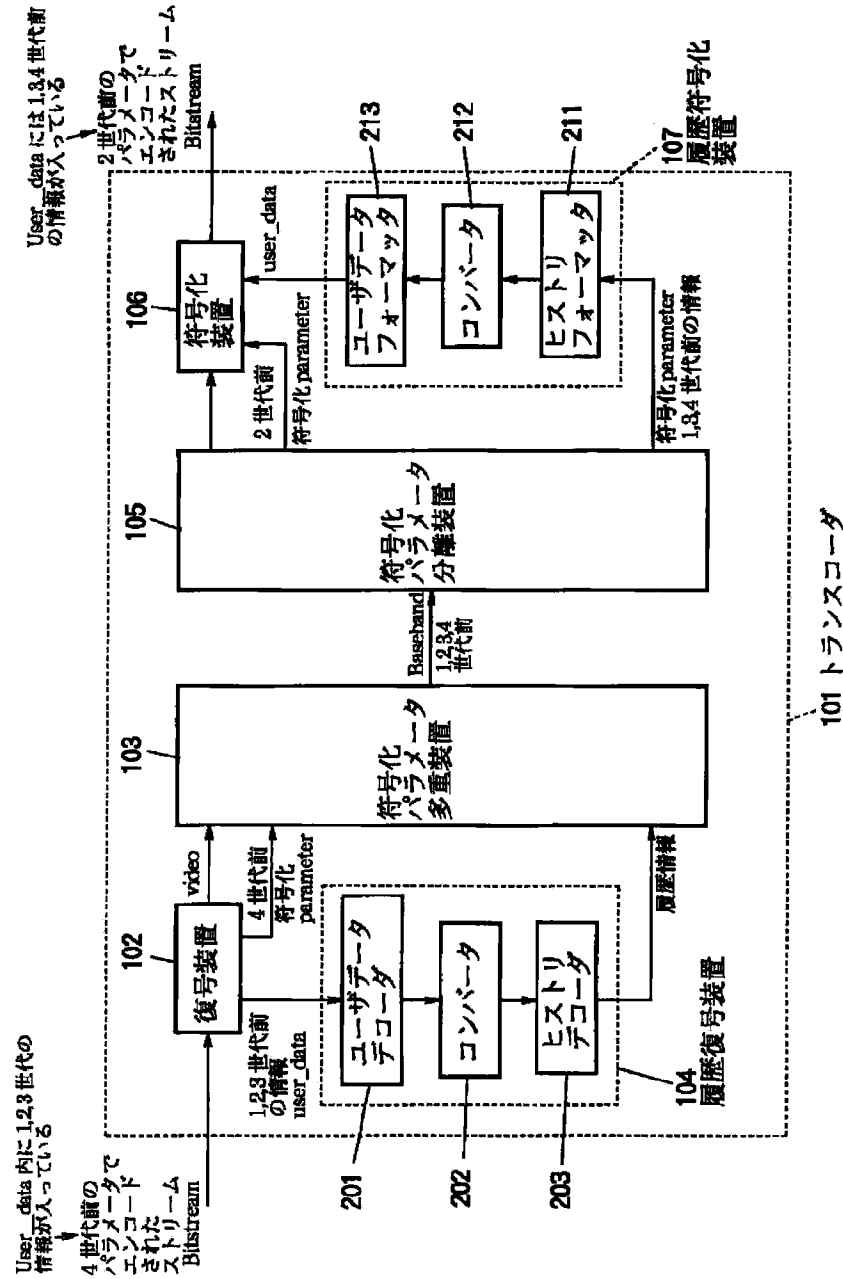


【図14】

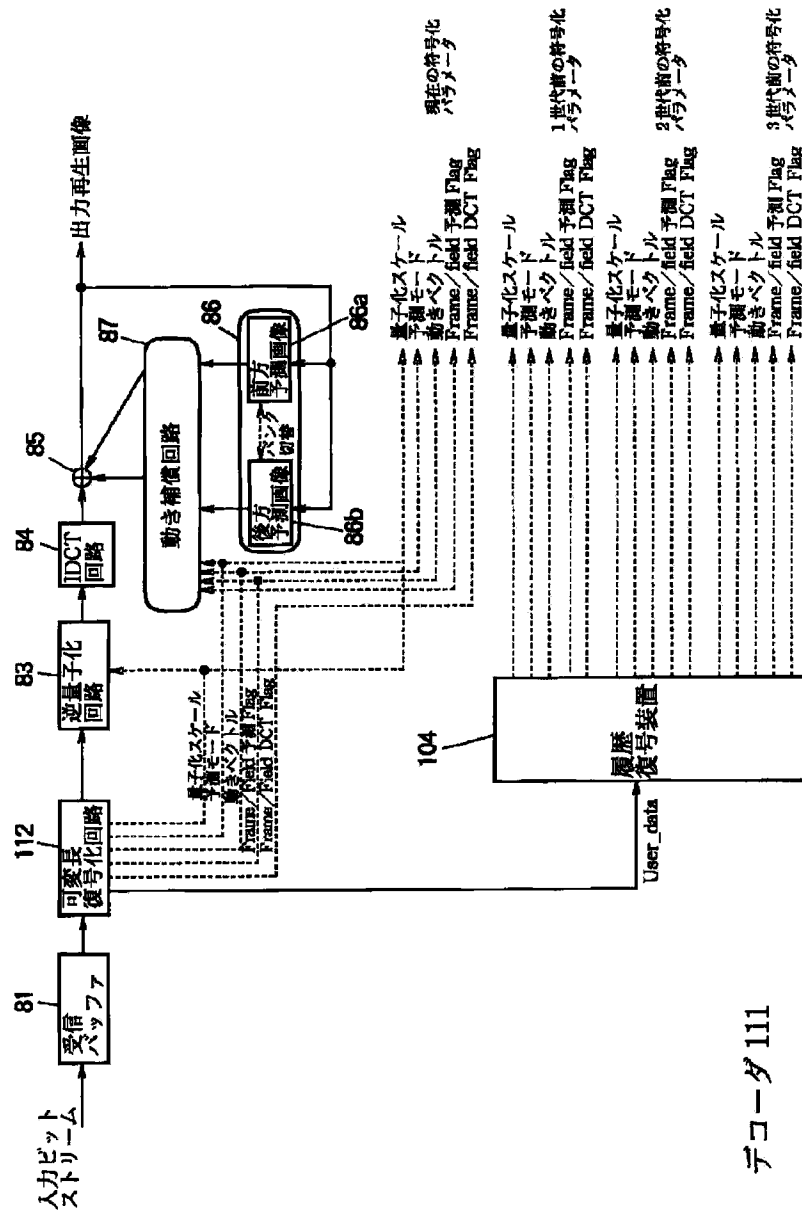


トランスコーダ 101

【図15】



【図16】



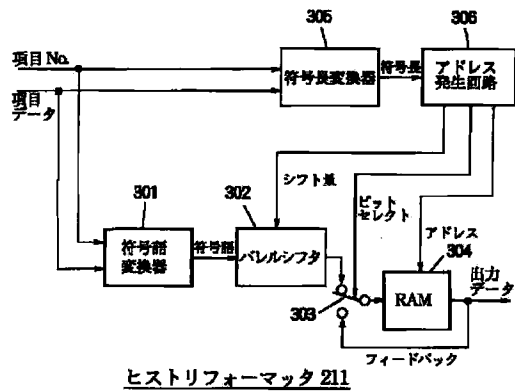
【図18】

区画								
D9	Cb[0][9]	Y[0][9]	Cr[0][9]	Y[1][9]	Cb[1][9]	Y[2][9]	Cr[1][9]	Y[3][9]
D8	Cb[0][8]	Y[0][8]	Cr[0][8]	Y[1][8]	Cb[1][8]	Y[2][8]	Cr[1][8]	Y[3][8]
D7	Cb[0][7]	Y[0][7]	Cr[0][7]	Y[1][7]	Cb[1][7]	Y[2][7]	Cr[1][7]	Y[3][7]
D6	Cb[0][6]	Y[0][6]	Cr[0][6]	Y[1][6]	Cb[1][6]	Y[2][6]	Cr[1][6]	Y[3][6]
D5	Cb[0][5]	Y[0][5]	Cr[0][5]	Y[1][5]	Cb[1][5]	Y[2][5]	Cr[1][5]	Y[3][5]
D4	Cb[0][4]	Y[0][4]	Cr[0][4]	Y[1][4]	Cb[1][4]	Y[2][4]	Cr[1][4]	Y[3][4]
D3	Cb[0][3]	Y[0][3]	Cr[0][3]	Y[1][3]	Cb[1][3]	Y[2][3]	Cr[1][3]	Y[3][3]
D2	Cb[0][2]	Y[0][2]	Cr[0][2]	Y[1][2]	Cb[1][2]	Y[2][2]	Cr[1][2]	Y[3][2]
D1	1世代前		2世代前		3世代前		最新	
D0	Cb[0][x]	Y[0][x]	Cr[0][x]	Y[1][x]	Cb[1][x]	Y[2][x]	Cr[1][x]	Y[3][x]

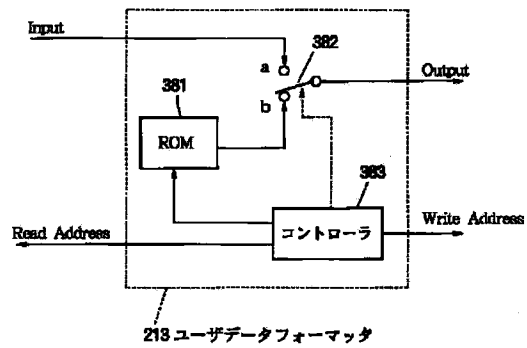
画像データ領域

符号化ハマータ領域

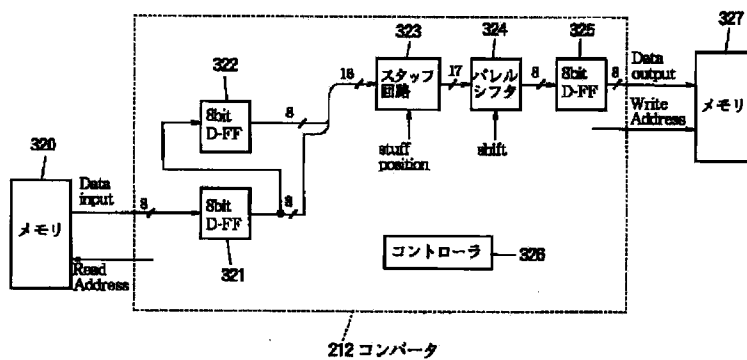
【図20】



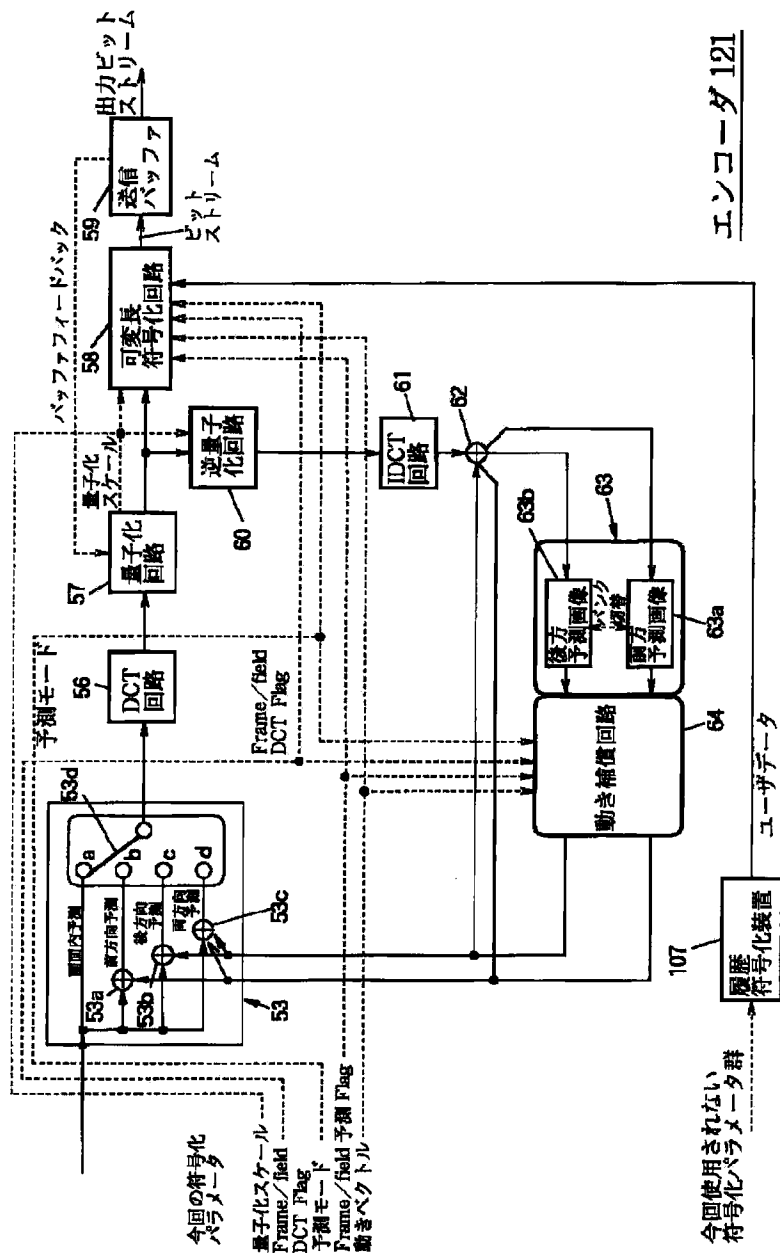
【図29】



【図22】

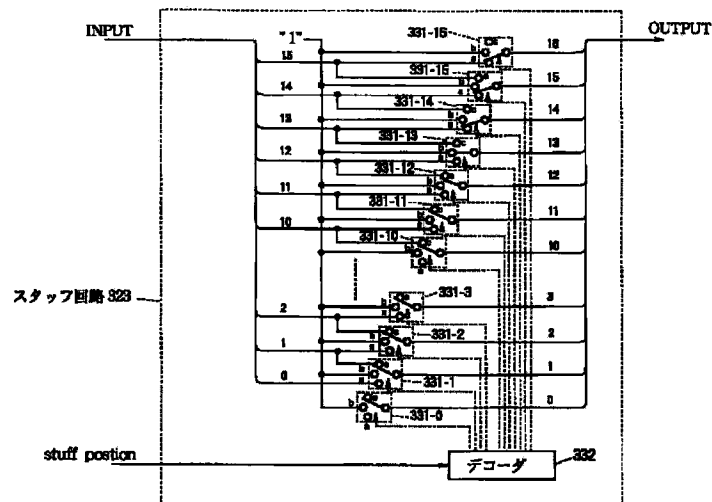


【図19】

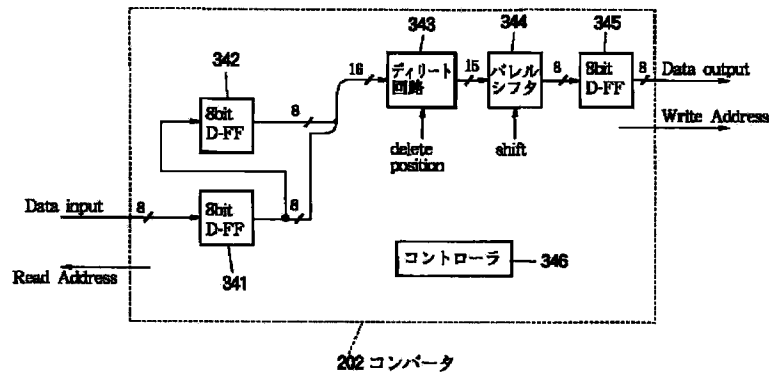


エンコーダ 121

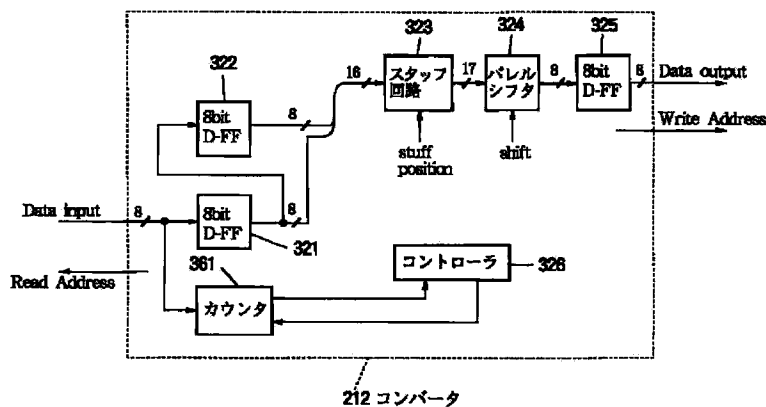
【図23】



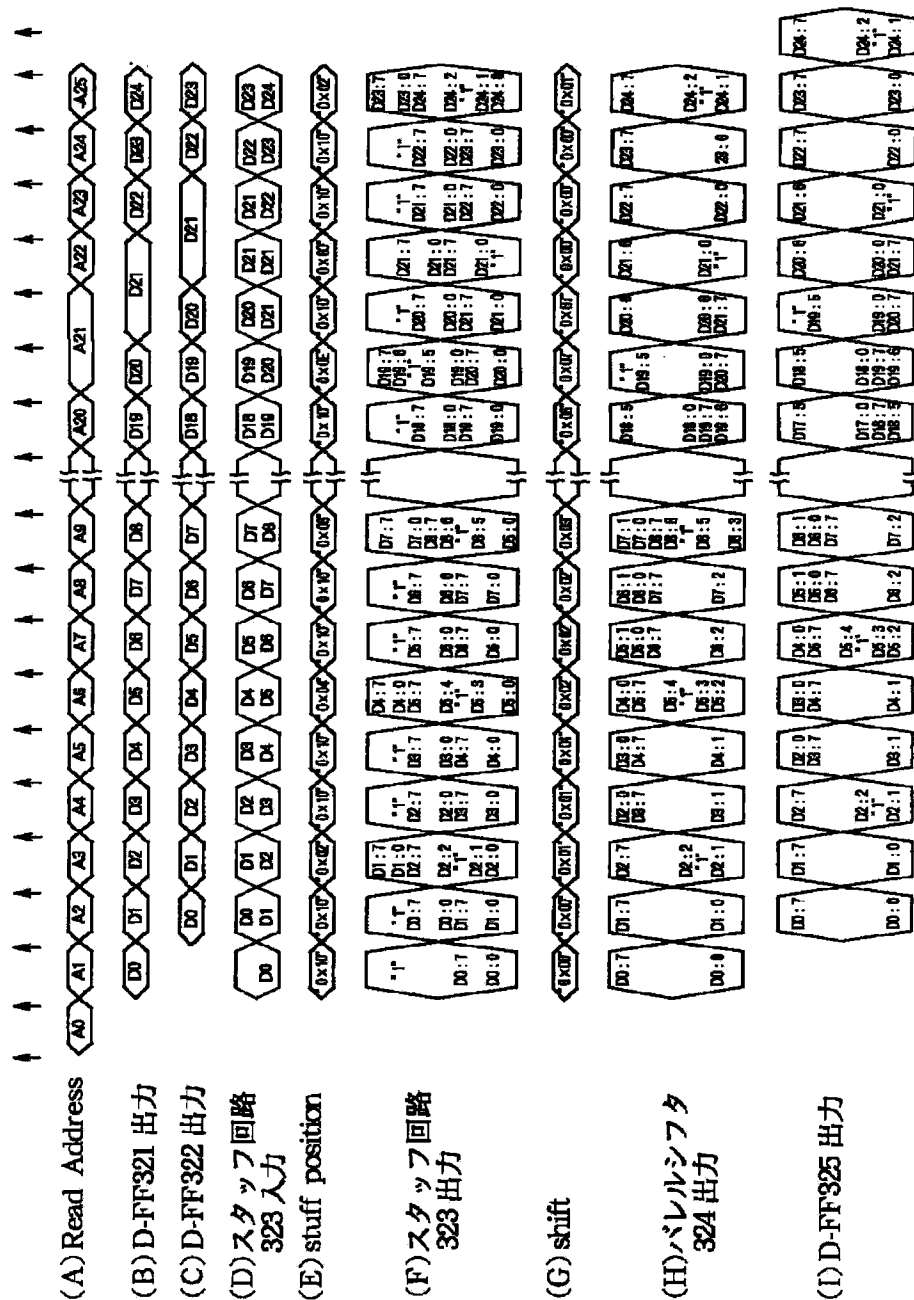
【図25】



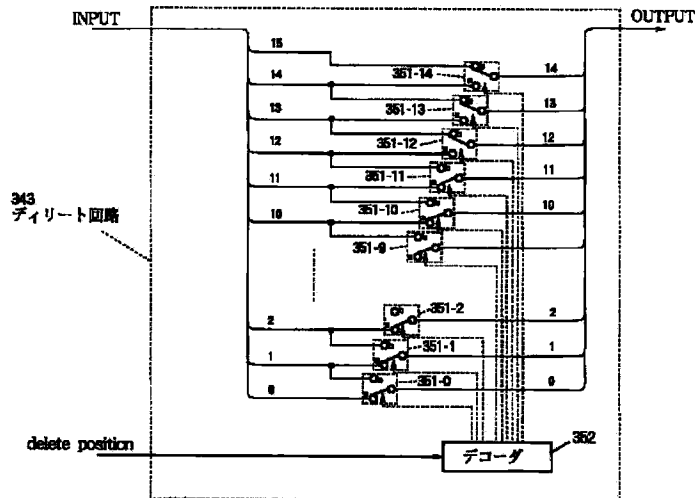
【図27】



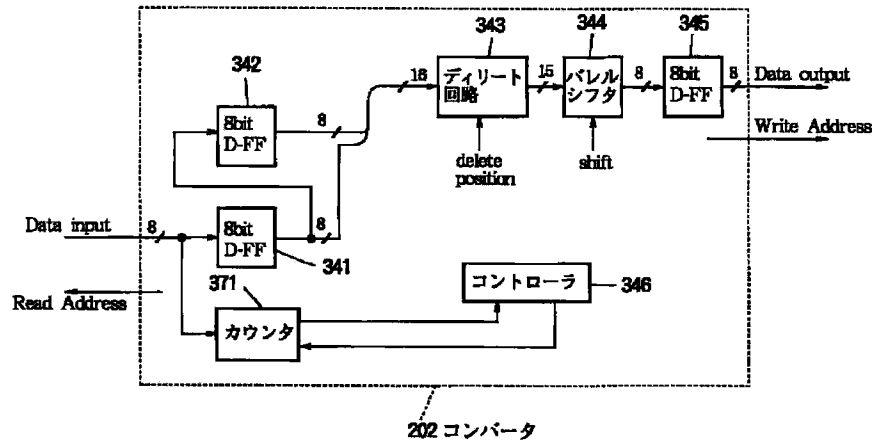
【図24】



【図26】



【図28】



【図35】

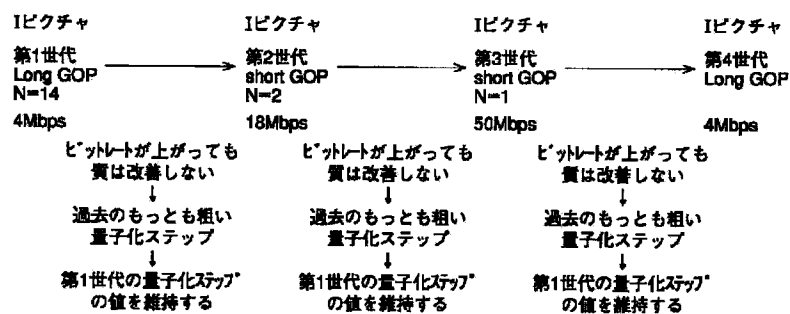
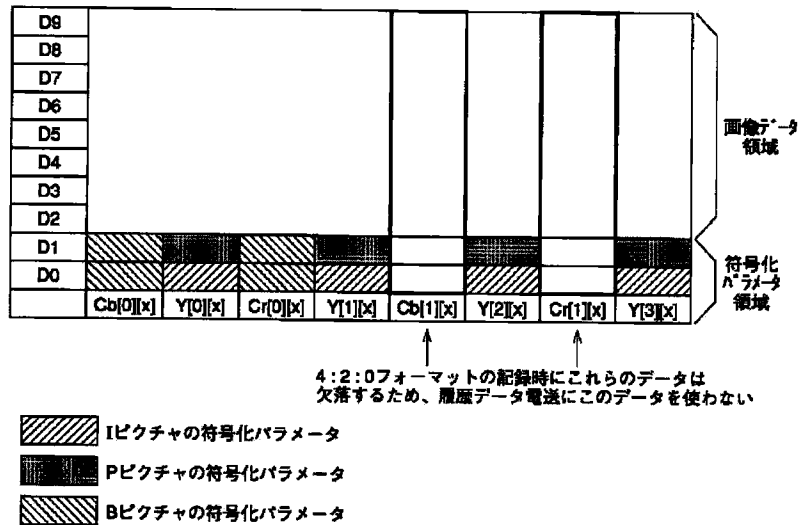
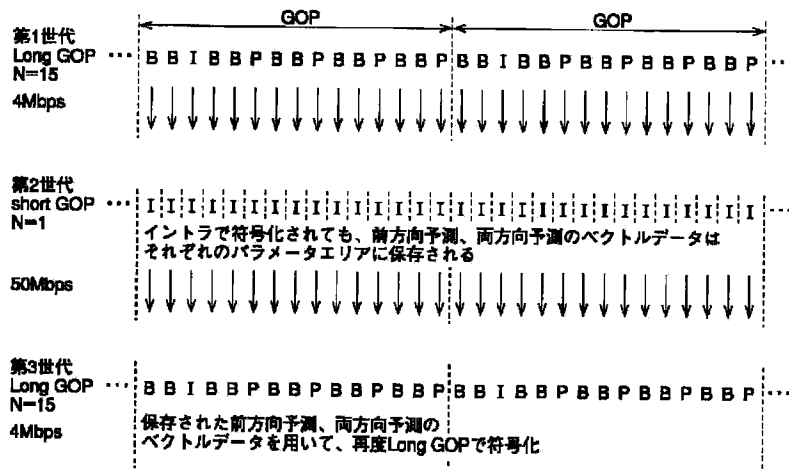


Figure 1 illustrates three types of video transcoders, labeled 第1世代 (1st Generation), 第2世代 (2nd Generation), and 第N世代 (Nth Generation). Each transcoder consists of a main processing block (101-1, 101-2, 101-N) and an SDTI output (108-1, 108-2, 108-N). The main processing block is divided into several sub-blocks: 102-1, 103-1, 105-1, and 106-1 for the 1st generation; 102-2, 103-2, 105-2, and 106-2 for the 2nd generation; and 102-N, 103-N, 105-N, and 106-N for the Nth generation. The sub-blocks are interconnected to process video data, including a video input, a video output, and a video signal path. The SDTI output is connected to the main processing block via a video signal path.

【图 3 1】



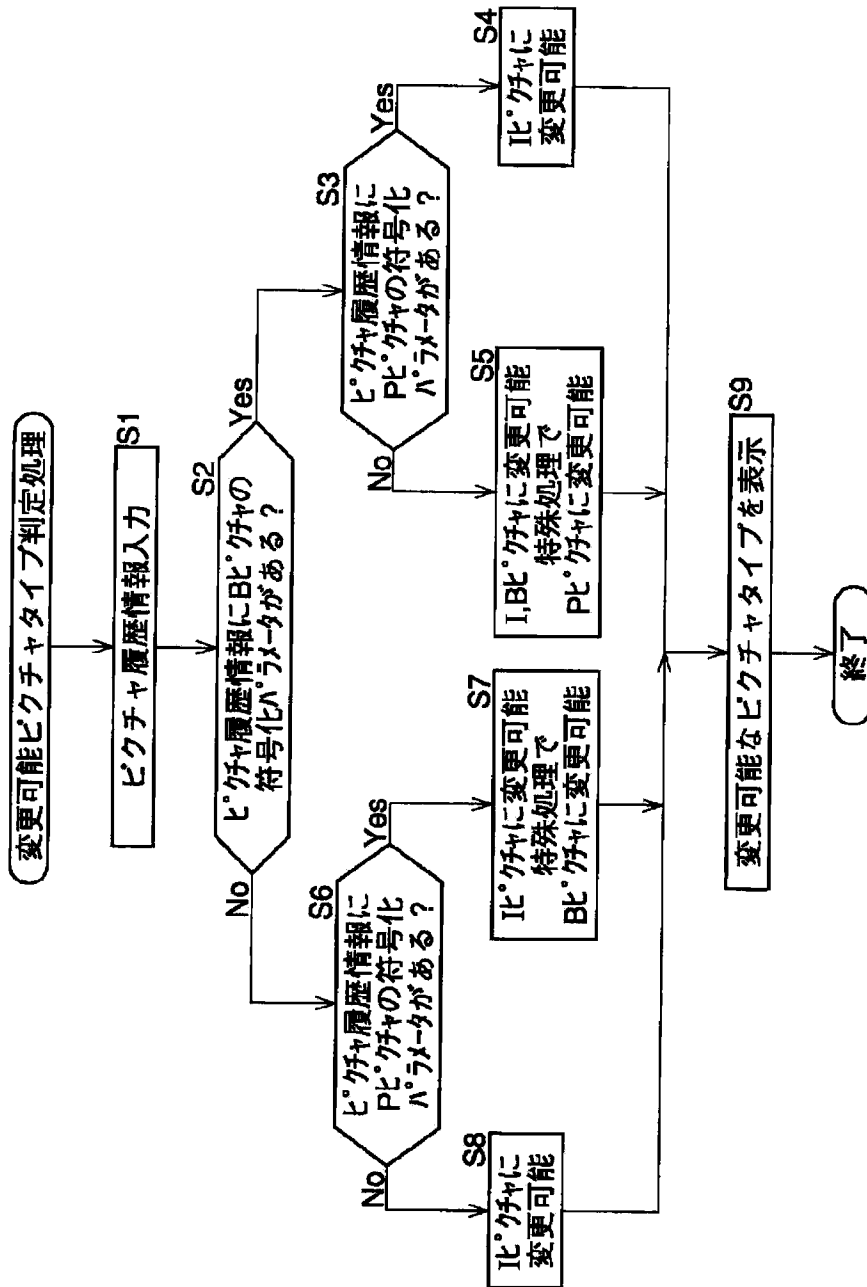
【図 3 3】



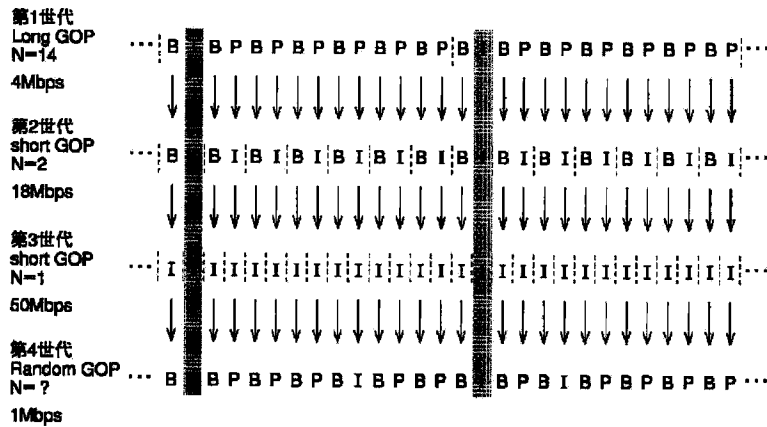
【図50】

extension_and_user_data(i) {	No. of bits	Mnemonic
while ((nextbits() == extension_start_code)		
(nextbits() == user_data_start_code)) {		
if ((i == 2) && (nextbits() == extension_start_code))		
extension_data()		
if (nextbits() == user_data_start_code)		
user_data()		
}		
}		

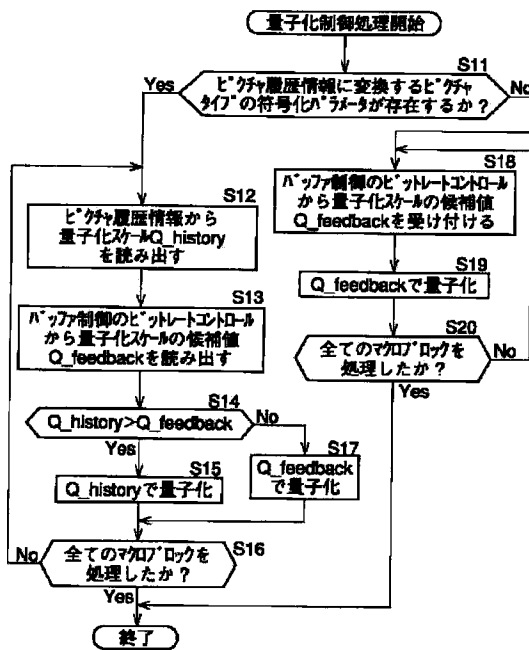
【図32】



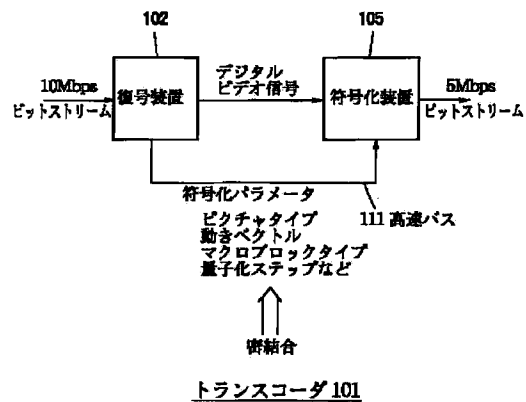
【図34】



【図36】



【図37】



【図51】

user_data() {	No. of bits	Mnemonic
user_data_start_code	32	balbf
while(nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	uimsbf
}		
next_start_code()		
}		

【図 38】

stream with history data

video_sequence() {	No. of bits	Mnemonic
next_start_code()		
sequence_header()		
sequence_extension()		
do {		
extension_and_user_data(0)		
do {		
if (nextbits() == group_start_code) {		
group_of_pictures_header(1)		
extension_and_user_data(1)		
}		
picture_header()		
picture_coding_extension()		
while ((nextbits() == extension_start_code)		
(nextbits() == user_data_start_code)) {		
if (nextbits() == extension_start_code)		
extension_data(2)		
if (nextbits() == user_data_start_code) {		
user_data_start_code	32	bslbf
if (nextbits() == History_Data_ID) {		
History_Data_ID	8	bslbf
converted_history_stream()		
}		
} else {		
user_data()		
}		
}		
picture_data()		
} while ((nextbits() == picture_start_code)		
(nextbits() == group_start_code))		
if (nextbits() != sequence_end_code) {		
sequence_header()		
sequence_extension()		
}		
} while (nextbits() != sequence_end_code)		
sequence_end_code	32	bslbf
}		

【図 59】

picture_data() {	No. of bits	Mnemonic
while (nextbits() == slice_start_code) {		
alice()		
}		
next_start_code()		
}		

【図40】

history stream(40-1)

history_stream() {	bits	value
sequence_header		
sequence_header_code	32	000001B3
sequence_header_present_flag	1	
horizontal_size_value	12	
marker_bit	1	1
vertical_size_value	12	
aspect_ratio_information	4	
frame_rate_code	4	
marker_bit	1	1
bit_rate_value	18	
marker_bit	1	1
vbv_buffer_size_value	10	
constrained_parameter_flag	1	0
load_intra_quantiser_matrix	1	
load_non_intra_quantiser_matrix	1	
marker_bits	5	1F
intra_quantiser_matrix[64]	8*64	
non_intra_quantiser_matrix[64]	8*64	
sequence_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	1
sequence_extension_present_flag	1	
profile_and_level_indication	8	
progressive_sequence	1	
chroma_format	2	
horizontal_size_extension	2	
vertical_size_extension	2	
marker_bit	1	1
bit_rate_extension	12	
vbv_buffer_size_extension	8	
low_delay	1	
marker_bit	1	1

【図41】

history stream(40-2)		
	bits	value
frame_rate_extension_n	2	
frame_rate_extension_d	5	
marker_bits	6	3F
sequence_display_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	2
sequence_display_extension_present_flag	1	
video_format	3	
colour_description	1	
colour_primaries	8	
transfer_characteristics	8	
marker_bit	1	1
matrix_coefficients	8	
display_horizontal_size	14	
marker_bit	1	1
display_vertical_size	14	
marker_bit	1	1
macroblock_assignment_in_user_data		
macroblock_assignment_present_flag	1	
marker_bits	7	7F
v_phase	8	
h_phase	8	
group_of_picture_header		
group_start_code	32	000001B8
group_of_picture_header_present_flag	1	
time_code	25	
closed_gop	1	
broken_link	1	
marker_bits	4	F
picture_header		
picture_start_code	32	00000100

【図52】

group_of_pictures_header() {	No. of bits	Mnemonic
group_start_code	32	balbf
time_code	25	balbf
closed_gop	1	uimsbf
broken_link	1	uimsbf
next_start_code()		
}		

【図42】

history stream(40-3)

	bits	value
temporal_reference	10	
picture_coding_type	3	
marker_bit	1	1
vbv_delay	16	
full_pel_forward_vector	1	
forward_f_code	3	
full_pel_backward_vector	1	
marker_bit	1	1
backward_f_code	3	
marker_bit	1	1
picture_coding_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	8
f_code[0][0]	4	
f_code[0][1]	4	
f_code[1][0]	4	
f_code[1][1]	4	
intra_dc_precision	2	
picture_structure	2	
top_field_first	1	
frame_pred_frame_dct	1	
concealment_motion_vectors	1	
q_scale_type	1	
marker_bit	1	1
intra_vlc_format	1	
alternate_scan	1	
repeat_first_field	1	
chroma_420_type	1	
progressive_frame	1	
composite_display_flag	1	
v_axis	1	
field_sequence	3	
sub_carrier	1	
burst_amplitude	7	

【図43】

history stream(40-4)

	bits	value
marker_bit	1	1
sub_carrier_phase	8	
quant_matrix_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	3
quant_matrix_extension_present_flag	1	
load_intra_quantiser_matrix	1	
marker_bits	2	3
intra_quantiser_matrix[64]	8*64	
load_non_intra_quantiser_matrix	1	
marker_bits	7	7F
non_intra_quantiser_matrix[64]	8*64	
load_chroma_intra_quantiser_matrix	1	
marker_bits	7	7F
chroma_intra_quantiser_matrix[64]	8*64	
load_chroma_non_intra_quantiser_matrix	1	
marker_bits	7	7F
chroma_non_intra_quantiser_matrix[64]	8*64	
copyright_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	4
copyright_extension_present_flag	1	
copyright_flag	1	
copyright_identifier	8	
original_or_copy	1	
marker_bit	1	
copyright_number_1	20	
marker_bit	1	
copyright_number_2	22	
marker_bit	1	
copyright_number_3	22	3F
marker_bits	6	

【図44】

history stream(40-5)

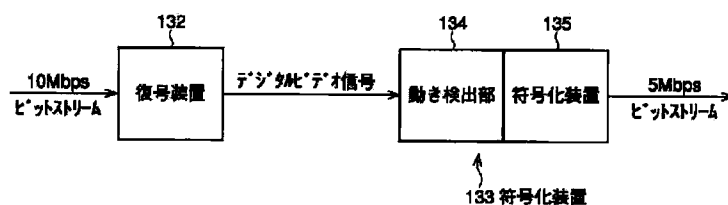
	bits	value
picture.display_extension		
extension_start_code	32	000001B5
extension_start_code_identifier	4	7
picture.display_extension_present_flag	1	
frame_centre_horizontal_offset_1	16	
marker_bit	1	1
frame_centre_vertical_offset_1	16	
marker_bit	1	1
frame_centre_horizontal_offset_2	16	
marker_bit	1	1
frame_centre_vertical_offset_2	16	
marker_bit	1	1
frame_centre_horizontal_offset_3	16	
marker_bit	1	1
frame_centre_vertical_offset_3	16	
marker_bits	6	3F
user_data		
user_data_start_code	32	000001B2
user_data	2048	
while(macroblock != macroblock_count){		
macroblock		
macroblock_address_h	8	
macroblock_address_v	8	
slice_header_present_flag	1	
skipped_macroblock_flag	1	
marker_bit	1	1
macroblock_modes()		
macroblock_quant	1	
macroblock_motion_forward	1	
macroblock_motion_backward	1	
macroblock_pattern	1	
macroblock_intra	1	

【図45】

history stream(40-6)

	bits	value
spatial_temporal_weight_code_flag	1	
frame_motion_type	2	
field_motion_type	2	
dct_type	1	
marker_bits	2	3
quantiser_scale_code	5	
marker_bits	3	7
PMV[0][0][0]	14	
marker_bits	2	3
PMV[0][0][1]	14	
motion_vertical_field_select[0][0]	1	
marker_bit	1	1
PMV[0][1][0]	14	
marker_bits	2	3
PMV[0][1][1]	14	
motion_vertical_field_select[0][1]	1	
marker_bit	1	1
PMV[1][0][0]	14	
marker_bits	2	3
PMV[1][0][1]	14	
motion_vertical_field_select[1][0]	1	
marker_bit	1	1
PMV[1][1][0]	14	
marker_bits	2	3
PMV[1][1][1]	14	
motion_vertical_field_select[1][1]	1	
marker_bit	1	1
coded_block_pattern	12	
marker_bits	4	F
num_mv_bits	8	
num_coef_bits	14	
marker_bits	2	3

【図69】



トランスコーダ 131

【図46】

history stream(40-7)

	bits	value
num_other_bits	7	
marker_bit	1	1
}		

【図47】

history_stream() {	No. of bits	Mnemonic
next_start_code()		
sequence_header()		
sequence_extension()		
extension_and_user_data(0)		
if (nextbits() == group_start_code) {		
group_of_pictures_header()		
extension_and_user_data(1)		
}		
picture_header()		
picture_coding_extension()		
extensions_and_user_data(2)		
picture_data()		
sequence_end_code	32	balbf
}		

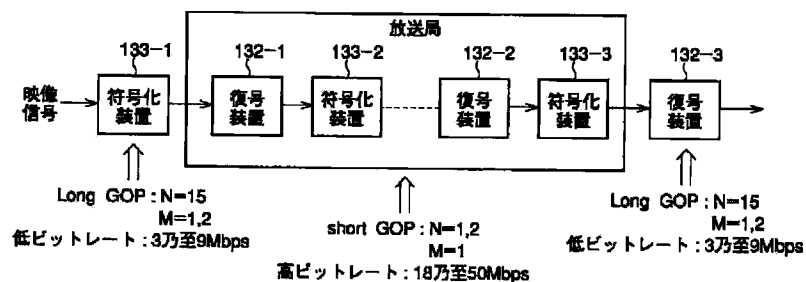
【図48】

sequence_header() {	No. of bits	Mnemonic
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
aspect_ratio_information	4	uimsbf
frame_rate_code	4	uimsbf
bit_rate_value	18	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_value	10	uimsbf
constrained_parameters_flag	1	bslbf
load_intra_quantiser_matrix	1	uimsbf
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8*64	uimsbf
load_non_intra_quantiser_matrix	1	uimsbf
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8*64	uimsbf
next_start_code()		
}		

【図49】

sequence_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
profile_and_level_indication	8	uimsbf
progressive_sequence	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	2	uimsbf
vertical_size_extension	2	uimsbf
bit_rate_extension	12	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_extension	8	uimsbf
low_delay	1	uimsbf
frame_rate_extension_n	2	uimsbf
frame_rate_extension_d	5	uimsbf
next_start_code()		
}		

【図70】



【図53】

picture.header() {	No. of bits	Mnemonic
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if (picture_coding_type == 2 picture_coding_type == 3) {		
full_pel_forward_vector	1	bslbf
forward_f_code	3	bslbf
}		
if (picture_coding_type == 3) {		
full_pel_backward_vector	1	bslbf
backward_f_code	3	bslbf
}		
while (nextbits() == '1') {		
extra_bit_picture /* with the value '1' */	1	uimsbf
extra_information_picture	8	uimsbf
}		
extra_bit_picture /* with the value '0' */	1	uimsbf
next_start_code()		
}		

【図55】

extension_data() {	No. of bits	Mnemonic
while (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
if (nextbits() == "Quant Matrix Extension ID")		
quant_matrix_extension()		
else if (nextbits() == "Copyright Extension ID")		
copyright_extension()		
else		
picture_display_extension()		
}		
}		

【図 5 4】

picture_coding_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
f_code[0][0] /* forward horizontal */	4	uimsbf
f_code[0][1] /* forward vertical */	4	uimsbf
f_code[1][0] /* backward horizontal */	4	uimsbf
f_code[1][1] /* backward vertical */	4	uimsbf
intra_dc_precision	2	uimsbf
picture_structure	2	uimsbf
top_field_first	1	uimsbf
frame_pred_frame_dct	1	uimsbf
concealment_motion_vectors	1	uimsbf
q_scale_type	1	uimsbf
intra_vlc_format	1	uimsbf
alternate_scan	1	uimsbf
repeat_first_field	1	uimsbf
chroma_420_type	1	uimsbf
progressive_frame	1	uimsbf
composite_display_flag	1	uimsbf
if (composite_display_flag) {		
v_axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	uimsbf
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
next_start_code()		
}		

【図 5 6】

quant_matrix_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
load_intra_quantiser_matrix	1	uimsbf
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8 * 64	uimsbf
load_non_intra_quantiser_matrix	1	uimsbf
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_intra_quantiser_matrix	1	uimsbf
if (load_chroma_intra_quantiser_matrix)		
chroma_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_non_intra_quantiser_matrix	1	uimsbf
if (load_chroma_non_intra_quantiser_matrix)		
chroma_non_intra_quantiser_matrix[64]	8 * 64	uimsbf
next_start_code()		
}		

【図57】

copyright_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
copyright_flag	1	bslbf
copyright_identifier	8	uimsbf
original_or_copy	1	bslbf
reserved	7	uimsbf
marker_bit	1	bslbf
copyright_number_1	20	uimsbf
marker_bit	1	bslbf
copyright_number_2	22	uimsbf
marker_bit	1	bslbf
copyright_number_3	22	uimsbf
next_start_code()		
}		

【図58】

picture_display_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
for (i=0; i<number_of_frame_centre_offsets; i++) {		
frame_centre_horizontal_offset	16	simsbf
marker_bit	1	bslbf
frame_centre_vertical_offset	16	simsbf
marker_bit	1	bslbf
}		
next_start_code()		
}		

【図63】

motion_vectors (s) {	No. of bits	Mnemonic
if (motion_vector_count == 1) {		
if ((mv_format == field) && (dmvs != 1))		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
} else {		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
motion_vertical_field_select[1][s]	1	uimsbf
motion_vector(1, s)		
}		
}		

【図60】

slice() {	No. of bits	Mnemonic
slice_start_code	32	bslbf
slice_quantiser_scale_code	5	uimsbf
if (nextbits() == '1') {		
intra_slice_flag	1	bslbf
intra_slice	1	uimsbf
reserved_bits	7	uimsbf
while (nextbits() == '1') {		
extra_bit_slice /* with the value '1' */	1	uimsbf
extra_information_slice	8	uimsbf
}		
extra_bit_slice /* with the value '0' */	1	uimsbf
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

【図61】

macroblock() {	No. of bits	Mnemonic
while (nextbits() == '0000 0001 000')		
macroblock_escape	11	bslbf
macroblock_address_increment	1-11	vlclbf
macroblock_modes()		
if (macroblock_quant)		
macroblock_quantiser_scale_code	5	uimsbf
if (macroblock_motion_forward		
(macroblock_intra && concealment_motion_vectors))		
motion_vectors(0)		
if (macroblock_motion_backward)		
motion_vectors(1)		
if (macroblock_intra && concealment_motion_vectors)		
marker_bit	1	bslbf
}		
}		

【図62】

macroblock_modes() {	No. of bits	Mnemonic
macroblock_type	1-9	vlcibf
if (macroblock_motion_forward macroblock_motion_backward) {		
if (picture_structure == 'frame') {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if ((picture_structure == "Frame picture") && (frame_pred_frame_dct == 0) && (dct_type_flag == 1)) {		
dct_type	1	uimsbf
}		
}		

【図64】

motion_vector (r, s) {	No. of bits	Mnemonic
motion_code[r][s][0]	1-11	vlcibf
if ((f_code[s][0] != 1) && (motion_code[r][s][0] != 0))		
motion_residual[r][s][0]	1-8	uimsbf
if (dmrv == 1)		
dmvector[0]	1-2	vlcibf
motion_code[r][s][1]	1-11	vlcibf
if ((f_code[s][1] != 1) && (motion_code[r][s][1] != 0))		
motion_residual[r][s][1]	1-8	uimsbf
if (dmrv == 1)		
dmvector[1]	1-2	vlcibf
}		

【図65】

macroblock_type VLC code					
		macroblock_quant			
				dct_type_flag	
				macroblock_motion_forward	
				macroblock_motion_backward	
				Description	
1	0	1	0	0	Intra
01	1	1	0	0	Intra, Quant

【図66】

macroblock_type VLC code					
		macroblock_quant			
				dct_type_flag	
				macroblock_motion_forward	
				macroblock_motion_backward	
				Description	
1	0	1	1	0	MC, Coded
01	0	1	0	0	No MC, Coded
001	0	0	0	0	MC, Not Coded
0001 1	0	1	0	0	Intra
0001 0	1	1	1	0	MC, Coded, Quant
0000 1	1	1	0	0	No MC, Coded, Quant
0000 01	1	1	0	0	Intra, Quant

【図67】

macroblock_type VLC code						
		macroblock_quant				
				dct_type_flag		
				macroblock_motion_forward		
				macroblock_motion_backward		
				Description		
10		0	0	0	0	Interp, Not Coded
11		0	1	1	1	Interp, Coded
010		0	0	0	0	Bwd, Not Coded
011		0	1	0	1	Bwd, Coded
0010		0	0	0	0	Fwd, Not Coded
0011		0	1	1	0	Fwd, Coded
0001 1		0	1	0	0	Intra
0001 0		1	1	1	1	Interp, Coded, Quant
0000 11		1	1	1	0	Fwd, Coded, Quant
0000 10		1	1	0	1	Bwd, Coded, Quant
0000 01		1	1	0	0	Intra, Quant

フロントページの続き

(72)発明者 三原 寛司
 東京都品川区北品川6丁目7番35号 ソニ
 ー株式会社内
 (72)発明者 村上 芳弘
 東京都品川区北品川6丁目7番35号 ソニ
 ー株式会社内

Fターム(参考) 5C059 MA00 MA05 MA23 MC38 PP05
 PP06 PP07 PP16 RB08 RC11
 RC31 SS01 SS07 SS11 SS20
 SS22 TA25 TA45 TB03 TB04
 TC15 TC20 TC41 UA02 UA05
 UA39